PM

Modelação de Serviços Afetos à Loja do Munícipe da CMF e Implementação de Protótipo de Portal Online

PROJETO DE MESTRADO

Maria José de Olim Gonçalves
MESTRADO EM ENGENHARIA INFORMÁTICA

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

fevereiro | 2019

Modelação de Serviços Afetos à Loja do Munícipe da CMF e Implementação de Protótipo de Portal Online

PROJETO DE MESTRADO

Maria José de Olim Gonçalves
MESTRADO EM ENGENHARIA INFORMÁTICA

ORIENTAÇÃO
David Sardinha Andrade de Aveiro

UNIVERSIDADE da MADEIRA

Modelação de Serviços afetos à Loja do Munícipe da CMF e Implementação de Protótipo de Portal Online

Maria José de Olim Gonçalves

Constituição do júri de provas públicas:

Karolina Baras, Prof.ª Auxiliar da Universidade da Madeira, Presidente

Pedro Campos, Prof. Auxiliar da Universidade da Madeira, Vogal

David Aveiro, Prof. Auxiliar da Universidade da Madeira, Vogal

Fevereiro 2019

Funchal - Portugal

Resumo

À medida que a sociedade e a tecnologia se tornam mais complexos, o mesmo acontece com as organizações. Uma vez que as organizações estão em constante expansão e evolução, é cada vez mais problemático entendê-las e garantir que os seus objetivos pretendidos sejam alcançados.

Muitas disciplinas têm ajudado a contribuir para a construção e manutenção das organizações, contudo não são suficientes para dominar todos os problemas complexos que estas enfrentam, o que leva à necessidade de ter uma visão mais ampla das organizações. Deste modo surge a disciplina de Engenharia Organizacional e a metodologia DEMO (*Design Engineering Methodology for Organizations*), cujo objetivo é criar modelos que ajudem a compreender a essência completa de uma organização.

No âmbito deste projeto os conceitos de Engenharia Organizacional e da metodologia DEMO são utilizados para modelar serviços afetos à Loja do Munícipe da Câmara Municipal do Funchal, com o propósito de auxiliar a construir modelos coerentes onde a competência, responsabilidade e autoridade estão bem definidas, evitando assim que ocorram anomalias nos processos.

Posteriormente à modelação foi desenvolvido um protótipo de portal online dinâmico e adaptável com vista ao suporte de modelação de diagramas e geração/execução automática de software.

Os componentes do protótipo foram desenvolvidos juntamente com dois colegas de mestrado e um bolseiro de investigação, e atualmente fazem parte de um projeto mais vasto do M-ITI (*Madeira Interactive Technologies Institute*).

Palavras-Chave

Engenharia Organizacional, DEMO, Protótipo, Software dinâmico e adaptável.

Abstract

As society and technology become more complex, so do organizations. Since organizations are constantly expanding and evolving, it is increasingly problematic to understand them and ensure that their intended goals are met.

Many disciplines have helped contribute to building and maintaining organizations, but they are not enough to address all the complex problems they face, which leads to the need for a broader view of organizations. In this way the Organizational Engineering discipline and the DEMO (*Design Engineering Methodology for Organizations*) methodology arises, whose objective is to create models that help to understand the complete essence of an organization.

Within the scope of this project, the concepts of Organizational Engineering and the DEMO methodology are used to model services related to the Municipality of Funchal, to help build coherent models where competence, responsibility and authority are well defined, thus avoiding anomalies occur.

After the modeling, a dynamic and adaptable online prototype portal was developed to support diagrams modeling and automatic software generation/execution.

Prototype's components were developed together with two masters collegues and a researcher, and are currently part of a larger project of the M-ITI (Madeira Interactive Technologies Institute).

Keywords

Enterprise Engineering, DEMO, Prototype, Dynamic and adaptable software.

Agradecimentos

Antes de mais, gostaria de agradecer ao meu orientador professor Doutor David Sardinha Andrade de Aveiro, pela orientação, apoio, paciência, disponibilidade e pelos conhecimentos que me transmitiu ao longo do desenvolvimento deste projeto.

Agradeço também aos colaboradores do Departamento de Sistemas de Informação e à Divisão de Jardins e Espaços Verdes Urbanos da Câmara Municipal do Funchal, pelo apoio e informações prestadas.

Não podia deixar de agradecer à minha família pela compreensão nas minhas várias ausências, pela força, incentivo e pela forma como sempre me apoiaram em todos os aspetos da minha vida.

Ao meu namorado pela paciência, motivação e por estar sempre do meu lado.

Aos meus colegas e amigos que me acompanharam neste percurso, aos utilizadores que testaram o protótipo, pelo tempo despendido e pelos conselhos, e a todos aqueles que direta ou indiretamente contribuíram para a realização deste projeto.

A todos expresso o meu sincero agradecimento por toda a ajuda prestada.

Índice

Lista de tabelas

Índice de figuras

Lista de siglas e acrónimos

AM (*Action Model)*

API (*Application Programming Interface)*

ARS (*Action Rule Specifications)*

ATD *(Actor Transaction Diagram)*

BCT (*Bank Contents Table)*

CDN (*Content Delivery Network*)

CM (*Construction Model)*

CSS (*Cascading Style Sheets*)

DDP (*Distributed Data Protocol*)

DEMO (*Design and Engineering Methodology for Organizations)*

DISME (*Dynamic Information System Modeler and Executer*)

HTTP (*HyperText Transfer Protocol*)

HTTPS *(Hyper Text Transfer Protocol Secure)*

IAM (*Interaction Model*)

IRS (Imposto sobre o Rendimento de Pessoas Singulares)

ISM (*Interstriction Model)*

JSON (*JavaScript Object Notation*)

LESS *(Leaner Style Sheets)*

M-ITI (*Madeira Interactive Technologies Institute*)

MVC (*Model View Controller*)

NPM *(Node Package Manager)*

OCD (*Organization Construction Diagram)*

OFD *(Object Fact Diagram)*

ORM (*Object-Relational Mapping*)

PDF (*Portable Document Format*)

PHP *(Hypertext Preprocessor)*

PIF *(Performa-Informa-Forma)*

PM (*Process Model)*

PSD *(Process Structure Diagram)*

PSI (*Performance in Social Interaction)*

SM (*State Model)*

SASS *(Syntactically Awesome StyleSheets)*

SQL (*Structured Query Language*)

TPD (*Transaction Pattern Diagram)*

TPT (*Transaction Product Table*)

TRT (*Transaction Result Table)*

URL *(Uniform Resource Locator)*

1 Introdução

O presente relatório destina-se a expor todo o trabalho desenvolvido ao longo do projeto de mestrado. Neste capítulo será descrito, resumidamente, as fases mais significativas do projeto, as principais motivações e objetivos, bem como a estrutura do relatório.

Na fase inicial do projeto, o objetivo passava pelo desenvolvimento de uma plataforma que incluía todos os requerimentos existentes na antiga plataforma da Câmara Municipal do Funchal, mas em formato digital. Com o intuito de implementar essa plataforma foram recolhidas algumas informações sobre os pedidos mais solicitados e foram elaborados protótipos do lado do utilizador e do lado do administrador. Quer a descrição dos pedidos quer os protótipos efetuados podem ser consultados no anexo A. Entretanto a meio do desenvolvimento dos protótipos, o objetivo inicial foi descontinuado e foi decidido proceder para um novo objetivo.

O novo objetivo passava pela extensão dos componentes de um protótipo desenvolvido por antigos alunos de licenciatura. Como esse projeto estava a ser desenvolvido em Wordpress, de modo a reutilizar os componentes, este projeto de mestrado também seria desenvolvido nas mesmas condições. Ainda foi efetuado algum trabalho com o Wordpress (disponível para consulta no anexo B), até que foi decidido proceder à implementação dos componentes juntamente com outros colegas de mestrado. Embora essa nova decisão não tivesse alterado o objetivo, foi definido que seria mais benéfico utilizar uma *framework*. A partir desse momento, o trabalho com o software Wordpress foi suspenso, e procedeu-se ao estudo e comparação de *frameworks*. Só após a escolha da *framework* mais indicada foi possível proceder à implementação dos componentes.

Atualmente, os componentes desenvolvidos neste projeto de mestrado inserem-se no contexto de um projeto mais vasto do M-ITI, denominado DISME (*Dynamic Information System Modeler and Executer*), que está a ser realizado de forma colaborativa com colegas de licenciatura e mestrado.

É de salientar que apesar dos objetivos deste projeto de mestrado terem sofrido algumas alterações, um dos objetivos que se manteve inalterado desde o início foi a utilização da metodologia DEMO para modelação dos processos da organização.

1.1 Motivação

Atualmente cerca de 75% dos projetos informáticos falham em cumprir satisfatoriamente as expetativas e as necessidades dos utilizadores [1] [2]. Uma das principais causas de tantas falhas nos projetos informáticos é precisamente uma compreensão incompleta e/ou ambígua da realidade organizacional a ser automatizada ou suportada por um sistema de informação, resultando em sistemas informáticos incompletos ou desadequados.

Além disso, os sistemas de informação e software em geral são, frequentemente, inflexíveis e complicados de modificar, sendo que é exigido um grande esforço para se acompanhar as sucessivas mudanças de requisitos e de legislação.

A disciplina de Engenharia Organizacional emergida da área de Sistemas de Informação nos anos 90, tem contribuído para solucionar estes problemas ao aplicar conceitos e métodos de engenharia às organizações com o objetivo de compreender e representar múltiplas facetas de uma organização, além de facilitar a análise e as mudanças organizacionais. Ou seja, a solução passa pelo desenvolvimento de teorias, métodos e ferramentas adequadas para uma análise e representação fidedigna da realidade estrutural e operacional de uma organização, seja pública ou privada, antes de se avançar para a automatização com base em Tecnologias de Informação [1].

1.2 Objetivos

Apesar da variação inicial de objetivos, neste subcapítulo são apresentados somente os objetivos definitivos do projeto.

O objetivo deste projeto passa inicialmente por realizar a modelação de serviços afetos à Loja do Munícipe da Câmara Municipal do Funchal aplicando a metodologia DEMO de Engenharia Organizacional e posteriormente pelo desenvolvimento de um protótipo de portal online.

O protótipo a ser desenvolvido destina-se ao suporte de modelação de diagramas e geração/execução automática de software e irá conter vários componentes que vão permitir a visualização e edição incremental de diagramas/modelos organizacionais, bem como a execução dos mesmos, através de um *dashboard*.

1.3 Estrutura do relatório

Este documento encontra-se dividido em seis capítulos. Neste primeiro capítulo "Introdução" é descrito resumidamente as fases mais significativas ao longo do desenvolvimento, é exposto a motivação para este projeto, bem como os seus objetivos.

Ao longo do capítulo dois "Introdução ao DEMO" são apresentados conceitos teóricos básicos referentes à metodologia DEMO que são essenciais para compreender todo o projeto.

O capítulo três "Cenários e aplicação de modelos" contém os passos efetuados na modelação dos casos, ou seja, é onde as análises, diagramas e tabelas elaboradas se encontram.

No capítulo quatro "Comparação de *frameworks*" são analisadas *frameworks* PHP (*Hypertext Preprocessor*) e Node.js e é decidido qual é a mais adequada para o desenvolvimento deste projeto.

Já o capítulo cinco "Desenvolvimento do protótipo" consiste em descrever todo o processo de desenvolvimento adotado. São apresentados os pontos mais importantes do protótipo, os requisitos funcionais e não funcionais, a arquitetura, as tecnologias utilizadas, os componentes desenvolvidos e respetivas funcionalidades.

Posteriormente, os testes de usabilidade efetuados com utilizadores são apresentados, tal como os seus resultados.

Por fim, no capítulo seis “Conclusões e trabalho futuro” são apresentadas as considerações finais sobre o DEMO e o protótipo desenvolvido e ainda são apresentadas melhorias para serem implementadas como trabalho futuro.

Posteriormente, as referências e os anexos são apresentados.

2 Introdução ao DEMO

A abordagem à ontologia de uma organização apresentada de seguida é a abordagem DEMO. DEMO é uma metodologia proveniente da disciplina de Engenharia Organizacional para o *design*, engenharia e implementação de organizações e está sustentada na teoria Ψ (lê-se *PSI - Performance in Social Interaction*), que tem como objetivo final capturar a essência completa de uma organização [2]. Seguindo a distinção feita nesta teoria entre as noções de função e construção de um sistema, dá-se o nome de *negócio da organização* ao conjunto de bens ou serviços que uma entidade fornece ao seu ambiente. Isto representa a perspetiva de função. Já às atividades coletivas de uma organização em que estes serviços são desempenhados ou entregues, incluindo os atores humanos que realizam estas atividades, é dado o nome de *estrutura da organização*, que por sua vez representa a perspetiva de construção.

As organizações são artefactos planeados e criados, tal como carros, aviões ou sistemas de informação. O que as diferencia é o facto dos seus elementos ativos serem seres humanos, mais concretamente os seres humanos na sua função social ou como sujeito [1].

2.1 Teoria Ψ

Para uma melhor compreensão do método DEMO, nesta secção será apresentado um breve resumo da teoria Ψ. A teoria Ψ é uma teoria sobre a construção e a operação das organizações e é constituída por quatro axiomas e um teorema que serão apresentados de seguida.

2.1.1 Axioma da Operação

O axioma da operação afirma que a operação de uma organização consiste em sujeitos que desempenham dois tipos de atos: atos de produção (*P-acts*) e atos de coordenação (C-a*cts*). Estes atos resultam, respetivamente, em factos de produção e factos de coordenação.

Ao desempenhar atos de produção, os sujeitos contribuem para a criação de bens ou serviços que são disponibilizados no ambiente. Os factos de produção resultantes podem ser divididos em atos materiais e atos imateriais. Os atos materiais referem-se a atos de criação, armazenamento e transporte de bens, enquanto que os atos imateriais se referem, por exemplo, a decisões e julgamentos.

Por outro lado, ao desempenhar atos de coordenação os sujeitos envolvem-se e cumprem compromissos mútuos com a finalidade de serem realizados atos de produção. Como exemplo de atos de coordenação tem-se o pedido, a promessa ou a rejeição [1].

Na figura 1 é apresentada uma representação gráfica do axioma de operação.

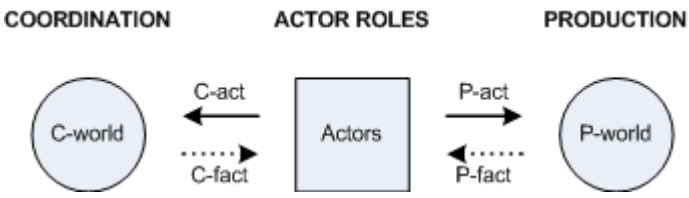


Figura 1 - Axioma da operação [1]

Como pode ser observado na figura 1, os atos de produção têm efeito no chamado mundo de produção (*P-World*), e os atos de coordenação têm efeito no mundo da coordenação (*C-World*). O estado de um mundo num determinado momento é o conjunto de factos criados até esse ponto do tempo. A criação de um facto de algum tipo é uma transição de estado em um dos dois mundos.

2.1.2 Axioma da Transação

O axioma da transação afirma que os atos de produção e os atos de coordenação ocorrem num padrão de coordenação genérico particular, ao qual é chamado de transação. Na figura 2, no canto superior direito, está retratado o padrão básico de transação como a formalização do fluxo de trabalho mais informal que está desenhado no canto superior esquerdo da mesma figura.

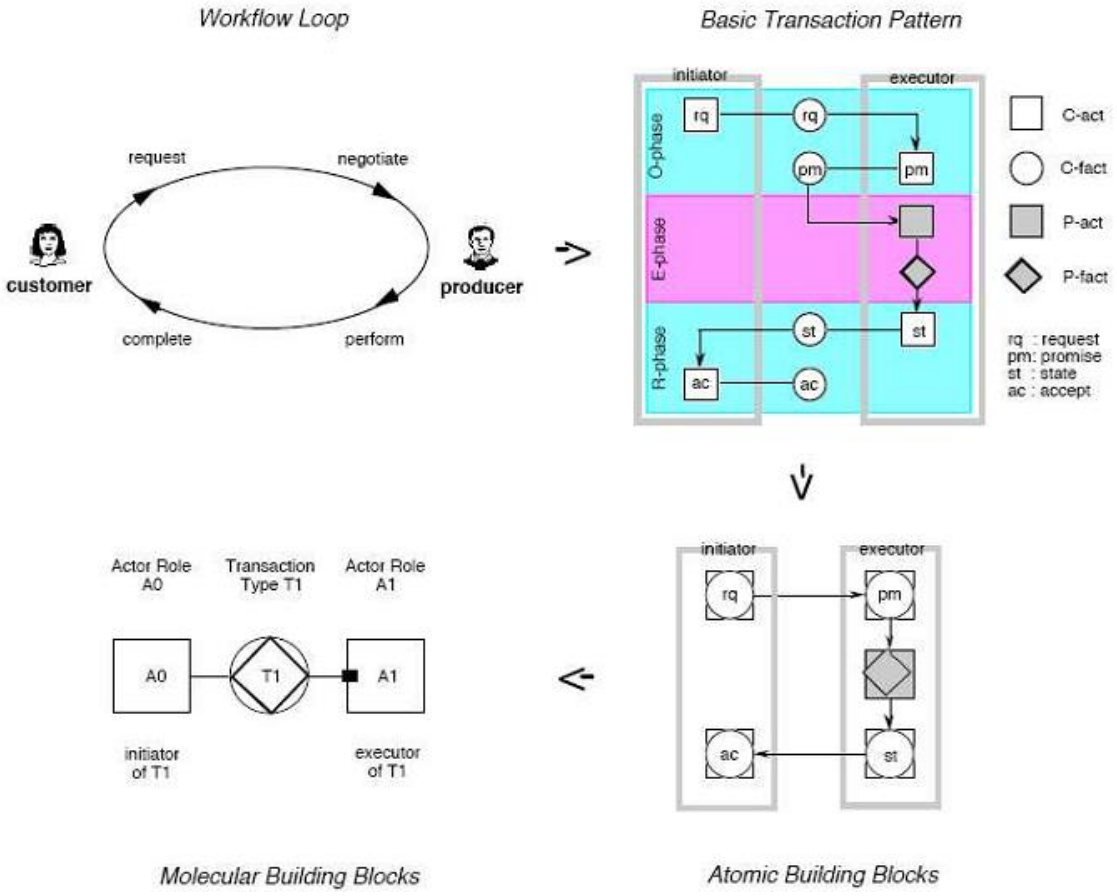


Figura 2 - Bloco da construção ontológica de uma organização [2]

Uma transação envolve dois atores, o iniciador e o executor. A transação especifica como os atos de coordenação e atos de produção se relacionam entre si, e

quem executa cada ato. Uma transação inclui um padrão de etapas bem definidas de coordenação e produção, sendo organizada em três fases: fase de pedido (*Order-phase*), fase de execução (*Execution-phase*) e fase de resultado (*Result-phase*).

Na fase de pedido, o iniciador e o executor negoceiam para chegar a um consenso sobre o resultado pretendido da transação, isto é, o facto de produção que o executor irá criar, bem como o tempo previsto de criação. Os principais atos de coordenação na fase do pedido são o pedido e a promessa. Na fase de execução, o executor produz um único facto de produção, e por fim na fase de resultado, o iniciador e o executor negoceiam para chegar a um acordo sobre o facto de produção produzido (que pode diferir do que foi realmente pedido). Os principais atos de coordenação na fase de resultado são a declaração e a aceitação.

Os termos "iniciador" e "executor" referem-se aos papéis de ator em vez dos sujeitos, e substituem os termos coloquiais de "cliente" e "produtor" respetivamente. Um papel de ator é definido como a autoridade e responsabilidade para ser o executor de um tipo de transação. Um sujeito pode desempenhar vários papéis de ator e um papel de ator pode ser desempenhado por vários sujeitos. Geralmente os papéis de ator não podem nem devem corresponder diretamente em nome nem em conteúdo a cargos ou funções organizacionais. Assim sendo, o nome destes papéis serão, sempre que possível, uma pequena variação do nome da transação executada. Por exemplo, num contexto de aluguer de carros, a transação *entrega de viatura* será executada pelo *entregador de viatura* (nome alternativo: responsável por entrega de viatura). Idealmente, um papel de ator deve ser desempenhado pelo mesmo sujeito para todos os atos sob a responsabilidade desse ator, nomeadamente o pedido e a aceitação pelo iniciador e a promessa e a declaração pelo executor. Deste modo, existe uma definição clara de responsabilidades que possibilita 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 ator a estas funções, e por sua vez, mapeamento de pessoas concretas às funções definidas.

O percurso real de uma transação pode ser mais complexo do que o padrão básico de transação que foi apresentado na figura 2 no canto superior direito. Isto é acomodado na teoria Ψ através de duas extensões formais ao padrão básico.

A primeira extensão refere-se aos casos em que os dois atores 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. Este processo de "declinar" ou "rejeitar" é chamado de estados de discussão, e poderá levar à paragem prematura de uma transação e eventuais cancelamentos de outros atos e/ou transações.

Em relação à segunda extensão, esta consiste em adicionar quatro padrões de cancelamento, um para cada um dos principais passos da transação, ou seja, para o pedido, promessa, declaração e aceitação.

Para uma melhor compreensão destes quatro padrões de cancelamento consideremos uma florista.

Padrão de cancelamento do pedido – Logo após pedir um *bouquet* de tulipas vermelhas, concluímos que as tulipas cor de rosa são mais bonitas, e dizemos à florista que afinal queremos as tulipas cor de rosa. A transação atual é cancelada e é iniciada uma nova transação.

Padrão de cancelamento da promessa – 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 ato do pedido. Não existe outra alternativa razoável a não ser permitir o cancelamento. Deste estado de cancelamento a escolha mais razoável é desistir, pois não faz sentido fazer o mesmo pedido novamente.

Padrão de cancelamento da declaração – Após 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. Neste momento podemos aceitar as tulipas e consequentemente aceitar o resultado da transação ou aceitar que a florista cancele a declaração e nos faça um novo *bouquet*.

Padrão de cancelamento da aceitação – Ao sair da florista começamos a sentir remorso, de que o *bouquet* não é exatamente como deveria ser. Voltamos a entrar e explicamos o nosso problema à florista, que o resolve, trocando o *bouquet* por outro (isto inclui o cancelamento do ato de declaração por parte da florista).

O padrão básico juntamente com estas duas extensões é denominado de padrão completo de transação e pode ser visualizado na figura 3.

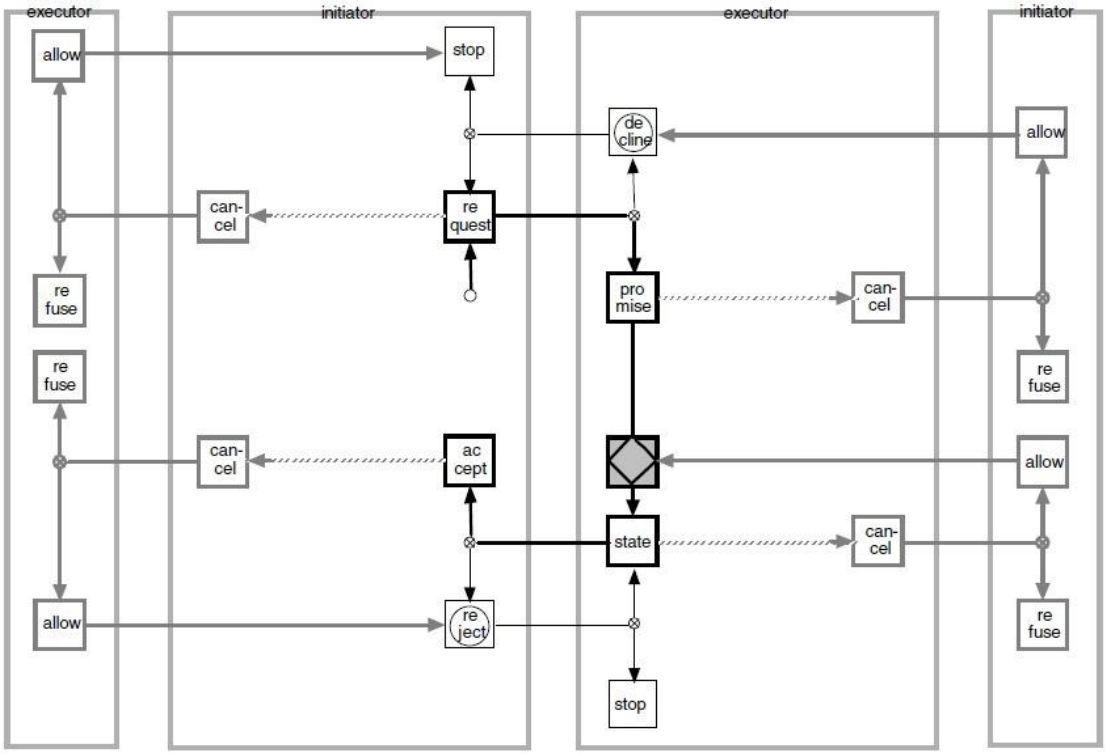


Figura 3 - Padrão completo de transação [2]

Este padrão, representado na figura 3, é considerado como uma lei socioeconómica: todas as transações em todos os tipos de organização seguem caminhos de atos ao longo deste padrão.

Realizar um ato de coordenação não implica necessariamente comunicação verbal ou escrita. Muitas vezes a promessa e a aceitação são realizadas pelo acenar ou por outro ato não-verbal. Por exemplo, numa padaria colocar o pão em cima do balcão em frente ao cliente conta como a realização da declaração. Para além disso os atos de coordenação podem ser efetuados tacitamente, o que significa que não existe qualquer ato que conte como a realização de um ato de coordenação. Os atos de coordenação tácitos devem ser compreendidos como concordância (implícita ou explícita) com o contrato que governa a transação. Um exemplo, na mesma padaria, é que a promessa pode ser efetuada sem que qualquer sinal seja observado.

Desempenhar um ato de coordenação tacitamente, contudo, não significa que este conte menos do que desempenhar esse mesmo ato de forma explícita. No entanto, para evitar ao máximo mal-entendidos, é aconselhável desempenhar todos os atos de coordenação de forma explícita. Uma das principais vantagens das tecnologias de informação e comunicação modernas é que o seu custo é bastante reduzido; no comércio eletrónico, como compras de artigos na internet, quase todos os atos de coordenação são efetuados explicitamente.

Na parte inferior direita da figura 2 é apresentada a notação composta do padrão de transação básico. Cada ato de coordenação e respetivo facto de coordenação resultante são representados por um símbolo composto. O mesmo também se aplica aos atos de produção e respetivos factos de produção.

Na parte inferior esquerda da figura 2, o padrão de transação completo é representado por um único símbolo, chamado de transação. Este símbolo consiste num diamante (representando a produção) embebido num disco (representando a coordenação). Os tipos de transação e papéis de ator são as unidades moleculares da construção de processos de negócio e de organização, sendo os passos da transação os átomos desta construção.

Para se chegar ao modelo ontológico de uma organização, a primeira abstração feita pela teoria Ψ é a aplicação do axioma da transação, que resulta numa enorme redução da complexidade, cerca de 70%, a nível de documentação [1].

2.1.3 Axioma da Composição

O axioma da composição explica como os factos de produção estão inter-relacionados. Este axioma afirma que as transações estão relacionadas umas com as outras de uma das duas formas possíveis [3]:

- Uma transação está incluída em outra transação;
- Uma transação é de auto ativação.

Segundo Jan L. G. Dietz, autor das teorias Engenharia Organizacional e do método DEMO, este axioma fornece a base para uma definição bem fundamentada da noção de processo de negócio:

"*Um processo de negócio é uma coleção de tipos de transações causalmente relacionadas, de modo que o passo de partida é um pedido executado por um papel de ator no ambiente (ativação externa) ou um pedido por um papel de ator interno para si próprio (auto ativação)"* [1].

2.1.4 Axioma da Distinção

Para compreender inteiramente a essência da operação das organizações, é apresentado o axioma da distinção que defende que os atores possuem três habilidades humanas, que são exercidas tanto em atos de coordenação como em atos de produção.

No que diz respeito à coordenação a habilidade *forma* refere-se ao formato em que a informação é transmitida (falar, ouvir), a habilidade *informa* relaciona-se com os aspetos do envolvimento do conteúdo (interpretar, formular) e a habilidade *performa* corresponde ao envolvimento num compromisso (expor ou invocar compromisso).

No que concerne à produção a habilidade *forma* diz respeito à produção documental ou datalógica (armazenar, copiar), a habilidade *informa* refere-se à produção informacional ou infológica (calcular, deduzir, computar) e a habilidade *performa* corresponde à produção essencial numa organização (julgar, decidir, conceber), e por isso também é designada de produção ontológica [1].

Na figura 4 está representado um resumo do Axioma da Distinção.

Figura 4 – Resumo do Axioma da Distinção (três habilidades humanas) [1]

Para que um ato de coordenação seja realizado com sucesso, os atos comunicativos nos três níveis devem ser realizados conforme a figura 5.

Para executar o ato de coordenação o *performer* (P) tem de expor o seu compromisso ao destinatário (A), ou seja, ambos devem chegar a um entendimento social. A única forma de conseguir esse entendimento é alcançar um entendimento intelectual (ou semântico). P tem de expressar corretamente os seus pensamentos, enquanto que A tem de deduzir esses pensamentos e confirmar que compreendeu (intelectualmente) a informação de P. P só pode expressar um pensamento,

formulando-o numa frase em algum idioma, e proferindo-o de alguma forma, como falar ou escrever. A, por outro lado, só pode deduzir o pensamento, ouvindo ou lendo. Portanto, é necessário um outro nível de entendimento, chamado de entendimento significacional (ou sintático). Por fim, é necessário existir uma troca física de dados através do campo da tecnologia e de tele(comunicações) [3].

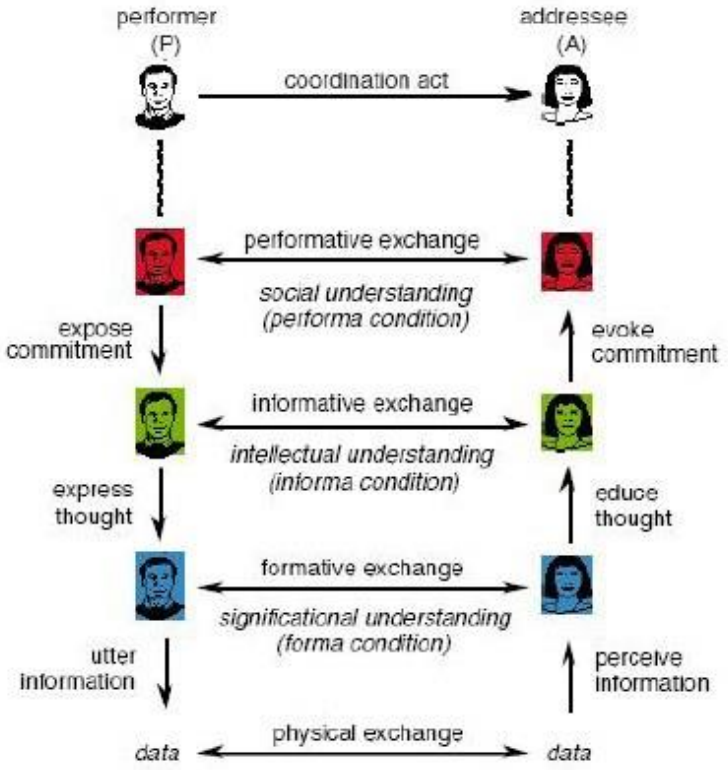


Figura 5 - Processo de execução de um ato de coordenação [1]

Com o axioma da distinção, a segunda abstração feita na teoria Ψ faz com que para se chegar ao modelo ontológico de uma organização apenas seja considerada a habilidade *performa* no que se refere à produção, abstraindo-se assim dos atos de produção a nível datalógico e infológico. Deste modo é conseguida uma segunda redução substancial de complexidade e diversidade, também estimada em cerca de 70% em termos de documentação.

Relativamente à coordenação e à produção, a habilidade *performa* é considerada a habilidade humana essencial para realizar negócios de qualquer tipo.

Os axiomas da teoria Ψ apresentados neste capítulo constituem poderosas ferramentas de abstração que possibilitam a especificação da operação e estrutura essencial das organizações, quer a nível de interações, como a nível de factos de negócio, independentemente da especificação ou não de todos os atos de cada transação 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 [1].

2.1.5 Teorema da Organização

O teorema da organização afirma que a organização de uma entidade é um sistema heterogéneo que é constituído pela integração em camadas de três sistemas homogéneos: a organização do negócio (*B-organization - Business*), a organização intelectual/informacional (*I-organization - Intellect/Information*) e a organização documental (*D-organization - Document*).

Todos os três sistemas pertencem à categoria de sistemas sociais, ou seja, as partes de coordenação são similares: os elementos são sujeitos que entram em acordo e cumprem compromissos uns para com os outros no que toca a atos de produção. Os sistemas apenas diferem no tipo de produção: a produção na organização de negócio é ontológica, a produção na organização intelectual/informacional é infológica e a produção na organização documental é datalógica.

Na figura 6 está representado o teorema da organização que demonstra como os três sistemas estão relacionados: a organização documental suporta a organização intelectual, e a organização intelectual suporta a organização de negócio.

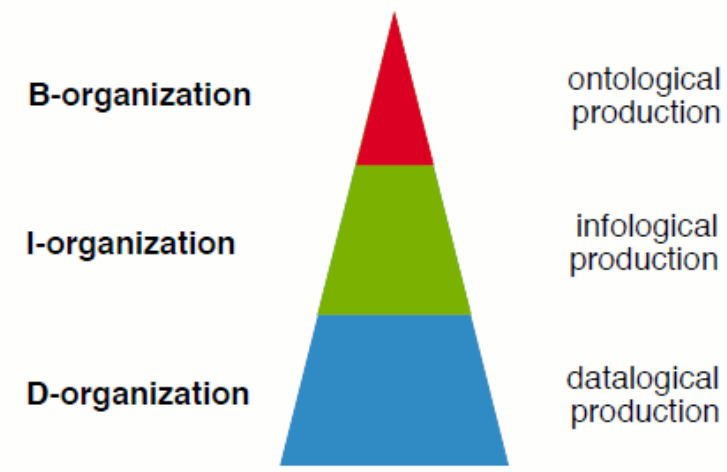


Figura 6 - Representação do Teorema da Organização [1]

As organizações de negócio, intelectual e documental consistem em sistemas que são facetas do todo da organização. A figura 7 apresenta de outra forma as camadas dessas três facetas. A distinção entre perspetiva de função (F) e de construção (C) tem como objetivo demonstrar como estas camadas estão relacionadas de uma forma mais clara.

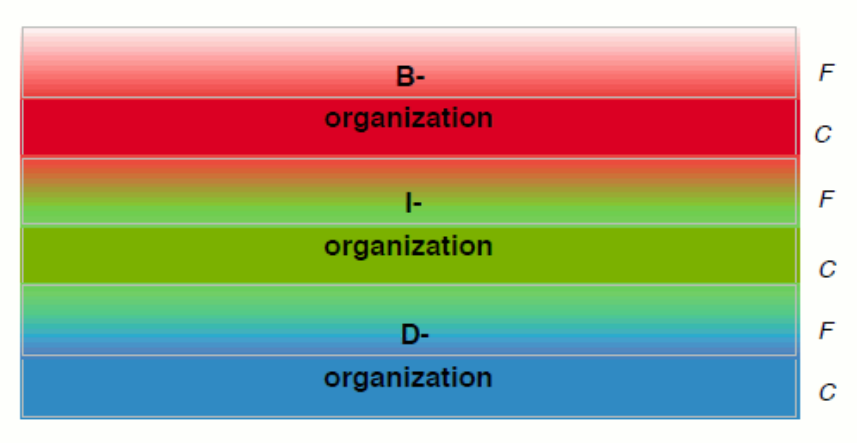


Figura 7 - Integração de uma Organização em camadas [1]

Como pode ser observado na figura 7 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 salientar a natureza intermediária da perspetiva de função a barra correspondente possui as cores da barra superior, inferior e a junção de ambas.

Concluindo, o Teorema da Organização combina os benefícios dos quatro axiomas mencionados anteriormente, numa noção coerente, concisa, abrangente e consistente da organização [1] [2].

2.2 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 consiste no conjunto integrado de quatro submodelos, em que cada um possui uma visão específica sobre a organização e podem ser representados por determinados diagramas, tabelas e listas.

Os quatro submodelos são os seguintes:

- Modelo de Construção (CM – *Construction Model*) – Constituído pelo Modelo de Interação (IAM - *Interaction Model*) e pelo Modelo de *Interstriction* (ISM - *Interstriction Model*);
- Modelo de Processo (PM – *Process Model*);
- Modelo de Estado (SM – *State Model*);
- Modelo de Ação (AM – *Action Model*).

Estes modelos, representados na figura 8, constituem o modelo ontológico completo da organização de negócio e, subsequentemente, representam o modelo ontológico da entidade correspondente. É de referir que o triângulo apresentado na figura 8 corresponde à parte superior do triângulo da figura 6.

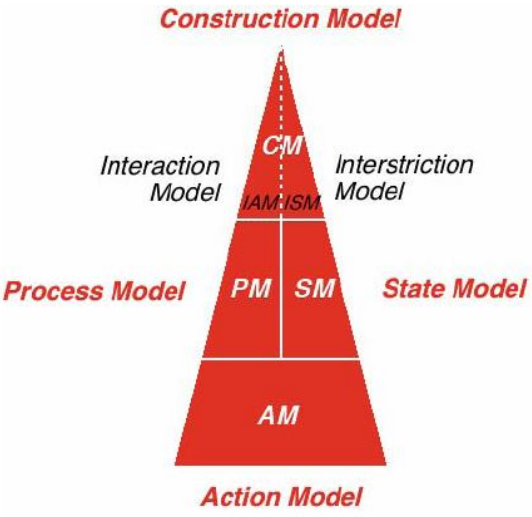


Figura 8 -Modelos DEMO de aspeto ontológico [2]

2.2.1 Modelo de Construção

O Modelo de Construção de uma organização é o modelo ontológico da sua construção. É o modelo mais conciso e especifica a construção do sistema organizacional, identificando os tipos de transações e os papéis de ator associados, tal como as ligações de informação entre os papéis de ator e os bancos de informação.

Este modelo está dividido em duas partes: o Modelo de Interação e o Modelo de *Interstriction*, em que o primeiro, representado pelo Diagrama de Atores e Transações (ATD - *Actor Transaction Diagram*), corresponde à parte ativa e o segundo corresponde à parte passiva. A estrutura de Interação de uma organização consiste nos tipos de transação em que os papéis de ator identificados participam como iniciador ou executor, enquanto que a estrutura *Interstriction* consiste nas ligações de informação entre os papéis de ator e bancos de informação que contêm os factos de negócio.

O Modelo de Construção de uma organização é expresso no Diagrama de Construção da Organização (OCD - *Organization Construction Diagram*), na Tabela de Produtos de Transação (TPT - *Transaction Product Table*) e na Tabela de Conteúdos do Banco (BCT - *Bank Contents Table*).

2.2.2 Modelo de Processo

O Modelo de Processos de uma organização é o modelo ontológico dos efeitos da sua operação no mundo da coordenação. O Modelo de Processo, inclui, para cada tipo de transação no Modelo de Construção, o padrão de transação específico do tipo de transação, isto é, contém os detalhes dos tipos de transações identificados.

O Modelo de Processo é representado pelo Diagrama de Estrutura De Processo (PSD - *Process Structure Diagram*), e é opcionalmente complementado por um Diagrama de Padrão de Transação (TPD - *Transaction Pattern Diagram*) para um ou mais tipos de transação.

No Diagrama de Estrutura De Processo estão representadas as relações causais (relações em que um ato de coordenação de uma transação provoca a ocorrência de um

ato de coordenação ou produção de outra transação) e as relações condicionais (relações em que um ato de coordenação ou produção de uma transação tem de esperar pela ocorrência de um ato de coordenação numa outra transação).

2.2.3 Modelo de Estado

O Modelo de Estado de uma organização especifica o espaço de estados e o espaço de transição do mundo de produção e contém classes de objetos, tipos de facto (objetos ou informações centrais de negócio), tipos de resultado (resultado da execução de transações, ou seja, factos de negócio afetando o estado de objetos anteriormente mencionados) e regras de existência (regras de unicidade e dependência de factos).

O Modelo de Estado é expresso no Diagrama de Factos e Objetos (OFD - *Object Fact Diagram*).

2.2.4 Modelo de Ação

O Modelo de Ação de uma organização contém as regras de ação que orientam os atores a lidar com a sua agenda. Existe, basicamente, uma regra de ação para cada papel de ator interno, e para cada tipo de *agendum* (ato de coordenação ocorrido) que os atores desse papel devem lidar.

O Modelo de Ação é representado através da Especificação das Regras de Ação (ARS - *Action Rule Specifications*).

Para resumir a descrição dos modelos efetuada anteriormente, é apresentada a figura 9, em que no lado esquerdo estão os diagramas em que os modelos são representados, e no lado direito estão as tabelas e respetivo cruzamento de informação entre os modelos [1] [2].

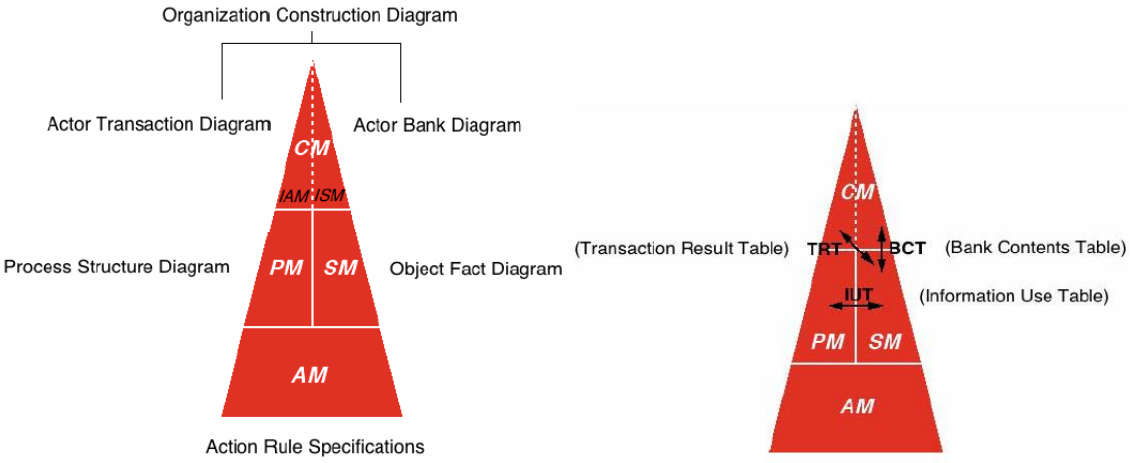


Figura 9 – Diagramas e Tabelas dos modelos DEMO [1]

3 Cenários de aplicação e modelos

Conhecendo os aspetos teóricos essenciais sobre a matéria podemos passar à prática. Com o objetivo de aplicar a metodologia DEMO, inicialmente foi efetuado o levantamento dos serviços da Câmara Municipal do Funchal que precisavam de ser modelados. Entre esses serviços foi decidido aplicar os métodos de engenharia organizacional ao caso "Funchal Cidade Florida" e ao caso de "Apoios para o desenvolvimento de atividades de interesse municipal".

Dispondo dos casos a modelar, o primeiro passo foi reunir toda a informação possível de qualquer tipo sobre os mesmos. Como essa informação se encontrava dispersa e não era suficiente para compreender todo o processo, foram elaboradas algumas questões com o intuito de complementá-la e posteriormente foi elaborado um documento contendo toda a informação recolhida sobre o caso.

Depois do documento estar concluído procedeu-se à análise PIF (*Performa-Informa-Forma*). A análise PIF consiste em colorir as habilidades *performa* de vermelho, as habilidades *informa* de verde e as habilidades *forma* de azul. Logo de seguida, os actos/factos de coordenação e os actos/resultados de produção foram agrupados em transações e posteriormente procedeu-se à identificação dos papéis de ator que foram destacados a amarelo.

Após esta análise avançou-se para a elaboração da TRT (*Transaction Result Table*). Na TRT estão presentes todas as transações detetadas e respetivos resultados, bem como o iniciador e o executor.

Depois da TRT estar concluída, passou-se à elaboração do ATD que é uma representação mais gráfica da TRT. Neste diagrama as transações são representadas por um disco com um símbolo de diamante no seu interior, que contém a respetiva combinação de atos de coordenação e atos de produção. Cada transação está conectada a dois retângulos que representam o ator iniciador e o ator executor. O iniciador está conectado ao símbolo de transação através de uma linha sólida simples, enquanto que os executores estão conectados ao símbolo de transação através de uma linha sólida que termina com um quadrado preto [4].

Uma vez concluído o ATD, produziu-se o PSD. No PSD possui-se uma visão mais detalhada dentro de cada transação, visto que é exibido em que momento (fase de ordem, execução ou resultado) e por que ordem as transações são desencadeadas [5]. Neste diagrama cada transação é representada por uma "salsicha" com o diamante de produção no seu interior. Para cada "salsicha" existe uma linha que separa o ator iniciador do executor, sendo que o iniciador se encontra na parte superior da linha e o executor na parte inferior. No PSD também estão presentes as ligações causais e ligações de espera. As ligações causais são representadas por uma seta sólida e indicam quais são os factos de outras transações que são causadas, enquanto que as ligações de espera são representadas por uma seta tracejada e indicam quais são os factos de precisam de aguardar para que um certo facto seja realizado. É de referir que na

produção do PSD, os padrões de desacordo e cancelamento não foram considerados, pelo que serão deixados para trabalho futuro.

Por último, o OFD foi concebido. No OFD são especificadas as classes de objetos, os tipos de facto, os tipos de resultado e ainda as leis existenciais [1]. As classes de objetos correspondem aos tipos de entidade e são representadas por uma "caixa" com as bordas arredondadas. Os tipos de facto são representados pelo símbolo que se encontra entre as classes de objetos, e os tipos de resultado são expressados através de um diamante que está conectado por uma linha sólida à classe de objeto correspondente. As leis que estão presentes no OFD elaborado são a lei da unicidade que é representada traçando uma linha na parte superior do tipo de facto, a lei da referência que é representada pela linha entre o tipo de facto e a classe de objeto e a lei da dependência que está representada pelo círculo a negrito na classe de objeto.

É de referir que apesar de existirem outros modelos, estes não serão aqui abordados visto que não foram relevantes para a implementação do protótipo.

Logo de seguida nos subcapítulos 3.1 e 3.2 são apresentados, respetivamente, os modelos elaborados referentes ao caso "Funchal Cidade Florida" e ao caso "Apoios para o desenvolvimento de atividades de interesse municipal".

3.1 Caso Funchal Cidade Florida

3.1.1 Análise PIF

O concurso Funchal Cidade Florida é promovido pela Câmara Municipal do Funchal, decorre entre os meses de Abril e Agosto e visa apelar ao envolvimento dos munícipes na melhoria e expansão da área de espaços verdes do Funchal e na promoção da agricultura urbana.

Todos os anos é elaborado uma proposta de deliberação pedindo a aprovação do concurso Funchal Cidade Florida {**T01** Request}. Essas propostas juntamente com o programa do concurso são levadas a reunião de Câmara pela [vereadora] em que a [assembleia] decide se o concurso deve ser aberto ou não {**T01** Execute}. Depois da aprovação pela [assembleia] {**T01** State}, o concurso é aberto {**T02** Execute} e compete ao [chefe da Divisão de Jardins e Espaços Verdes Urbanos], proceder à nomeação do júri {**T03** Execute}. No entanto a [vereadora] tem de aprovar essa nomeação de júri {**T04** Execute} efetuada pelo [chefe da Divisão].

O [júri] será quem decidirá a atribuição dos prémios {**T05** Execute} e deverá ser constituído por três elementos, com qualificações adequadas à análise dos pressupostos do concurso, sendo os mesmos membros da Câmara Municipal, membros do gabinete de apoio à presidência, do gabinete de apoio à vereação ou funcionários da Câmara Municipal.

Normalmente as inscrições decorrem entre 1 e 30 de Abril. No primeiro dia das inscrições a [funcionária da Divisão de Jardins e Espaços Verdes Urbanos], com responsabilidades relativas ao Concurso Funchal Cidade Florida, pede ao [Departamento de Informática] para atualizar a página web da Câmara Municipal do Funchal com

informações relativas à abertura do concurso {**T02** Request}, nomeadamente o programa do concurso, a ficha de inscrição e o período durante o qual as inscrições estão abertas.

Quando o concurso é aberto {**T02** Execute}, o [candidato] pode submeter o seu pedido de candidatura {**T06** Request} através do formulário que se encontra disponível no site da autarquia ou então pode entregá-lo presencialmente na Quinta do Poço da Câmara.

Após fecho da época de inscrições, a [funcionária da Divisão de Jardins e Espaços Verdes Urbanos] tem acesso a todas as candidaturas, quer submetidas on-line quer entregues presencialmente. Todas as candidaturas são analisadas pela funcionária que verifica se os dados estão todos preenchidos e se os dados inseridos pelo utilizador são os corretos (por exemplo se as moradas são válidas) para serem admitidos ao concurso {**T06** Execute}. Caso haja dados incorretos a [funcionária da Divisão de Jardins e Espaços Verdes Urbanos] contacta o [candidato] por telefone para fornecer as informações corretas, caso o contacto não seja válido, a candidatura é excluída.

É de referir que após a data limite das inscrições para o concurso, a [funcionária da Divisão de Jardins e Espaços Verdes Urbanos] contacta novamente o [Departamento de Informática] para atualizar as informações do site relativas ao concurso, nomeadamente para referir que as inscrições já não estão disponíveis e para retirar o programa do concurso.

A [funcionária da Divisão de Jardins e Espaços Verdes Urbanos] após reunir e organizar informações referentes a todas as candidaturas (online e presenciais) envia todos esses dados aos [membros do júri] para que estes possam deslocar-se até aos locais concorrentes para avaliá-los. Os [membros do júri] visitam todos os locais no mês de Junho, e durante a visita analisam os espaços e avaliam individualmente {**T07** Execute} (cada um dá a sua nota) considerando vários critérios tais como: resultado global do conjunto, natureza das espécies utilizadas, racionalidade da manutenção, interesse estético do espaço e funcionalidade do espaço planeado.

Posteriormente após terem visitado todos os espaços concorrentes, o [júri] reúne-se para proceder à apreciação final, ou seja, para decidir a classificação final dos candidatos {**T05** Execute}. Nessa reunião o [júri] elabora uma ata onde estão descritas várias informações como por exemplo o número total de participantes e em que modalidades se inserem e principalmente, é onde são apresentados os vencedores para cada modalidade e os valores dos prémios atribuídos. Para chegar à conclusão de quem é o vencedor são efetuados cálculos, sendo a nota final igual à média aritmética do somatório ponderado das pontuações para cada requisito, de cada um dos elementos do júri.

Seguidamente essa ata é enviada para a [Divisão de Jardins e Espaços Verdes Urbanos], e a [funcionária da Divisão de Jardins e Espaços Verdes Urbanos] encarrega-se de contactar por telefone os [participantes] vencedores a informar apenas em que dia, a que horas e onde a cerimónia de entrega de prémios vai acontecer.

Só na cerimónia de entrega de prémios é que os participantes têm conhecimento em que lugar ficaram, e que prémio ganharam. Também durante a cerimónia é realizada uma exposição com fotografias dos espaços vencedores.

Depois da cerimónia de prémios, novamente o site da autarquia é atualizado e são colocadas as fotos da exposição.

3.1.2 TRT

Transação	Resultado	Papel Iniciador	Função iniciadora	Papel executante	Função executante
T01 - Aprovação da abertura do concurso	R01 - Aprovação da abertura do concurso foi efetuada	Gestor de concursos	Divisão de Jardins e Espaços Verdes Urbanos	Aprovador da abertura do concurso	Assembleia
T02 - Abertura do concurso	R02 - Abertura do concurso foi efetuada	Gestor de concursos	Divisão de Jardins e Espaços Verdes Urbanos	Responsável por abrir concurso	Departamento de Informática
T03 - Nomeação do júri para concurso	R03 - Nomeação do júri para concurso foi efetuada	Gestor de concursos	Divisão de Jardins e Espaços Verdes Urbanos	Nomeador de Júri	Chefe da Divisão de Jardins e Espaços Verdes Urbanos
T04 - Aprovação de nomeação do júri para concurso	R04 - Aprovação de nomeação do júri foi efetuada	Gestor de concursos	Chefe da Divisão de Jardins e Espaços	Aprovador de nomeação de júri	Vereadora
T05 – Avaliação final de candidatura	R05 - Avaliação final de candidatura foi efetuada	Responsável por admissão de candidatura	Divisão de Jardins e Espaços Verdes Urbanos	Avaliador final de candidatura	Júri
T06 - Admissão de candidatura	R06 - Admissão de candidatura foi efetuada	Candidato	Munícipe	Responsável por admissão de candidatura	Divisão de Jardins e Espaços Verdes Urbanos
T07 – Pré-Avaliação de candidatura	R07 - Pré-Avaliação de candidatura foi efetuada	Responsável por admissão de candidatura	Divisão de Jardins e Espaços Verdes Urbanos	Pré-Avaliador de candidatura	Júri
T08- Gestão de Concurso	R08 - Gestão de Concursos foi efetuada	Gestor de concursos	Divisão de Jardins e Espaços Verdes Urbanos	Gestor de concursos	Divisão de Jardins e Espaços Verdes Urbanos

Tabela 1 - TRT caso Funchal Cidade Florida

3.1.3 ATD

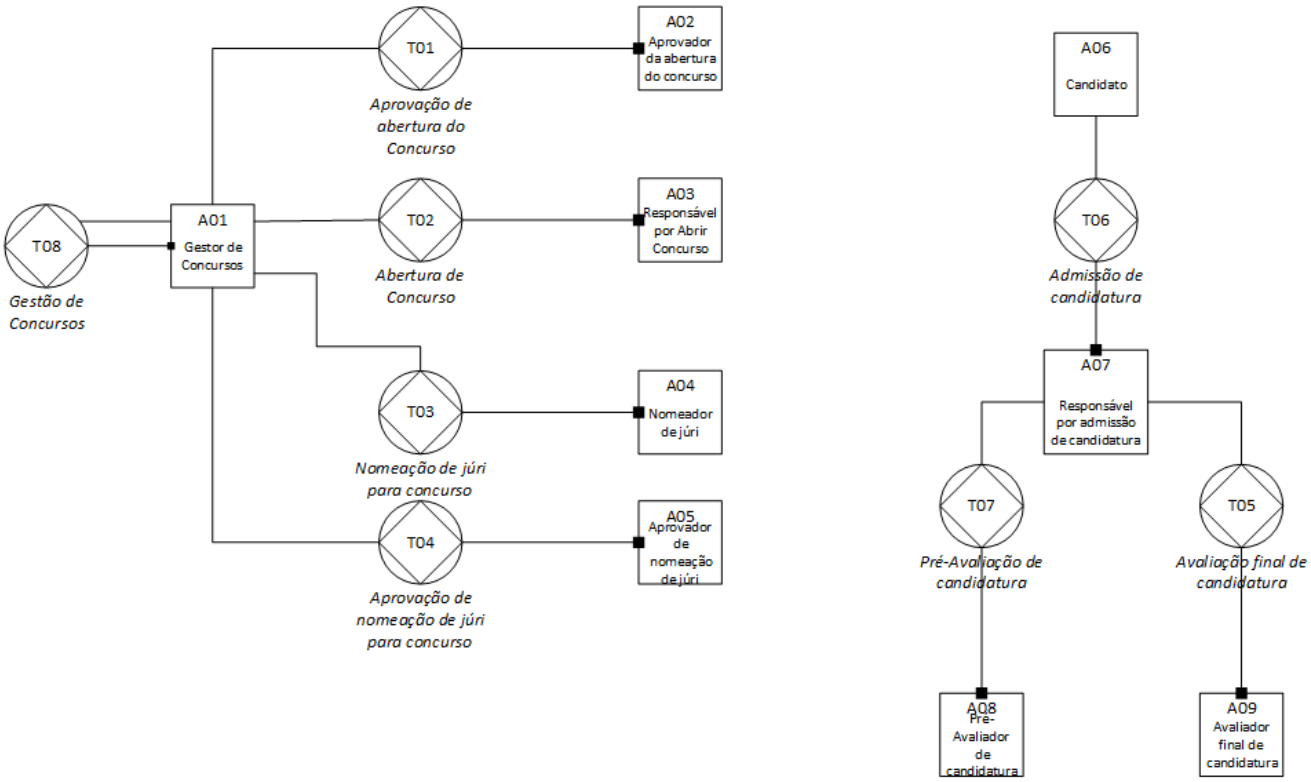


Figura 10 - ATD caso Funchal Cidade Florida

3.1.4 PSD

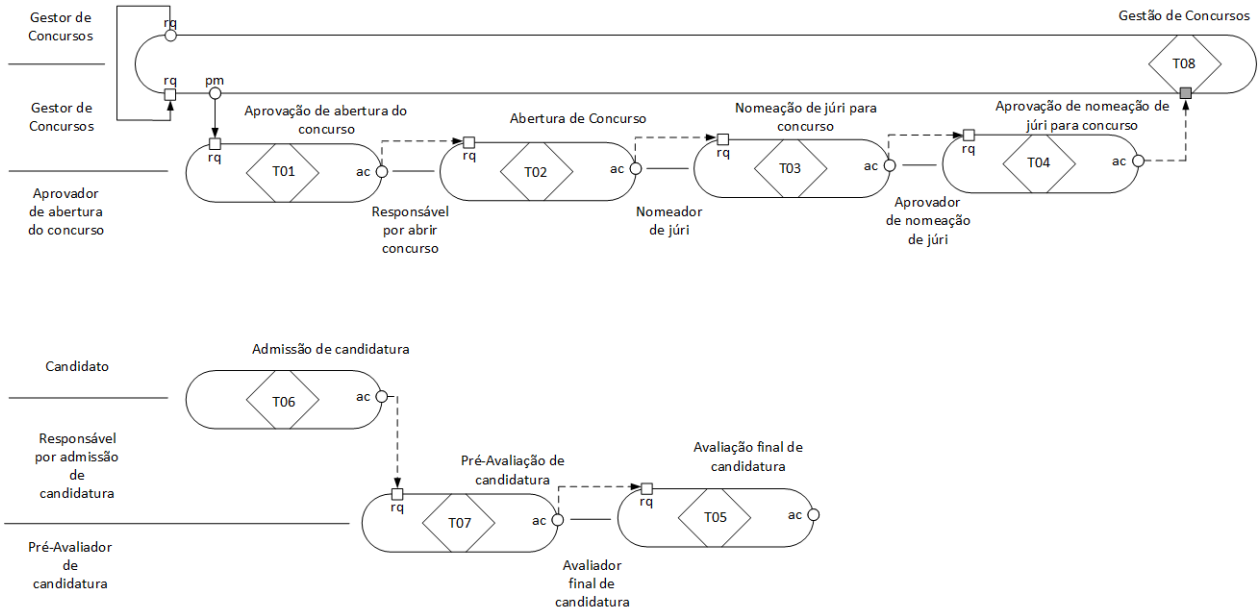


Figura 11 - PSD caso Funchal Cidade Florida

3.1.5 OFD

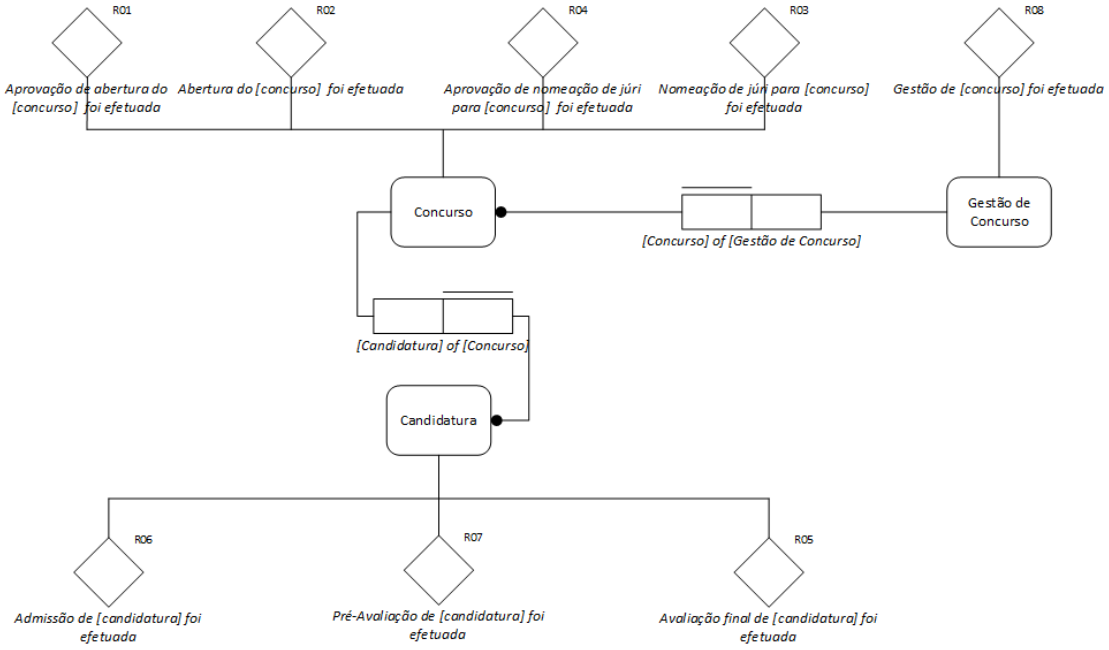


Figura 12 - OFD caso Funchal Cidade Florida

3.2 Caso de Apoios para o desenvolvimento de atividades de interesse municipal

3.2.1 Análise PIF

Os pedidos para candidatura {T01 **Request**} a apoios podem ser apresentados ao Município do Funchal entre 1 de Janeiro até ao dia 31 de Agosto do ano anterior ao da execução da atividade ou projeto pretendido. No entanto, este prazo pode ser dispensado nos pedidos de apoio a projetos e atividades cuja ocorrência não era expetável até à data ali estipulada, e podem ser apresentados ao município do Funchal a todo o tempo, desde que razões de interesse municipal devidamente fundamentadas o justifiquem.

A atribuição de apoios visa auxiliar a atividade das pessoas coletivas de fim altruístico, assim como promover o desenvolvimento de projetos e eventos em áreas de interesse municipal, tais como: Informação e defesa dos interesses dos cidadãos; Social; Cultural; Educativa; Desportiva; Recreativa; Ambiente e património natural; Promoção da saúde e prevenção de doenças; Promoção do desenvolvimento económico; Promoção da igualdade de género; Promoção da cidadania e dos direitos humanos e Proteção Civil.

Para este apoio podem candidatar-se [pessoas], [entidades] e [organismos legalmente existentes] que possuam sede ou não no Município do Funchal desde que desenvolvam atividades relevantes e no interesse da cidade do Funchal.

Para que uma candidatura possa ser admitida, o [participante] tem de submeter dados sobre o "Apoio Financeiro" {T02 **Request**}, "Dados Gerais" {T03 **Request**}, "Fundamentação do interesse das atividades para o município do Funchal" {T04 **Request**} e ainda um "Historial resumido da entidade" {T05 **Request**}.

Posteriormente, o [Departamento Financeiro] é que se responsabiliza por analisar as candidaturas e também se responsabiliza por separar as candidaturas pela área adequada conforme o seu tipo. Por exemplo, as candidaturas de carácter cultural são encaminhadas para o [Departamento e Economia e Cultura]; as de carácter Social são encaminhadas para o [Departamento Social]; as desportivas para o [Departamento de Educação e Qualidade de vida]; as de proteção civil para o [Serviço Municipal de Proteção Civil], etc.

Cada um desses [Departamentos] faz uma seleção das candidaturas {T06 **Execute**} e produz um relatório descrevendo quais são as candidaturas de maior interesse para a CMF e as quais vale a pena investir.

Seguidamente, esse relatório é enviado para a [CMF] e é efetuada uma reunião de Câmara com os 11 [vereadores] em que finalmente é decidido quais serão os apoios que serão cedidos {T07 **Execute**}.

Os apoios atribuídos serão publicitados anualmente no sítio oficial do Município do Funchal na internet, e ainda será enviado uma notificação às entidades ou pessoas que tenham pedido os apoios.

3.2.2 TRT

Transação	Resultado	Papel Iniciador	Função iniciadora	Papel executante	Função executante
T01 - Admissão de candidatura a apoios	R01 - Admissão de candidatura a apoios foi efetuada	Candidato a admissão de apoios	Munícipe	Admissor de candidatura a apoios	Departamento Financeiro
T02 - Submissão de dados de apoio financeiro	R02 - Submissão de dados de apoio financeiro foi efetuada	Admissor de candidatura a apoios	CMF	Submissor de dados de apoio financeiro	Munícipe
T03 – Submissão de dados gerais	R03 - Submissão de dados gerais foi efetuada	Admissor de candidatura a apoios	CMF	Submissor de dados gerais	Munícipe
T04 – Submissão do interesse das atividades para o município do Funchal	R04 - Submissão do interesse das atividades para o município do Funchal foi efetuada	Admissor de candidatura a apoios	CMF	Submissor do interesse das atividades para o município do Funchal	Munícipe
T05 – Submissão do historial da entidade	R05 - Submissão do historial da entidade foi efetuada	Admissor de candidatura a apoios	CMF	Submissor do historial da entidade	Munícipe
T06 – Seleção de candidatura para apoios	R06 - Seleção de candidatura para apoios foi efetuada	Admissor de candidatura a apoios	CMF	Selecionador de candidatura a apoios	Departamento responsável
T07 – Decisão sobre candidatura a apoios	R07 - Decisão sobre candidatura a apoios foi efetuada	Selecionador de candidatura a apoios	Departamento responsável	Decisor sobre atribuição de apoios	Presidência

Tabela 2 - TRT caso Apoios para o desenvolvimento de atividades de interesse municipal

3.2.3 ATD

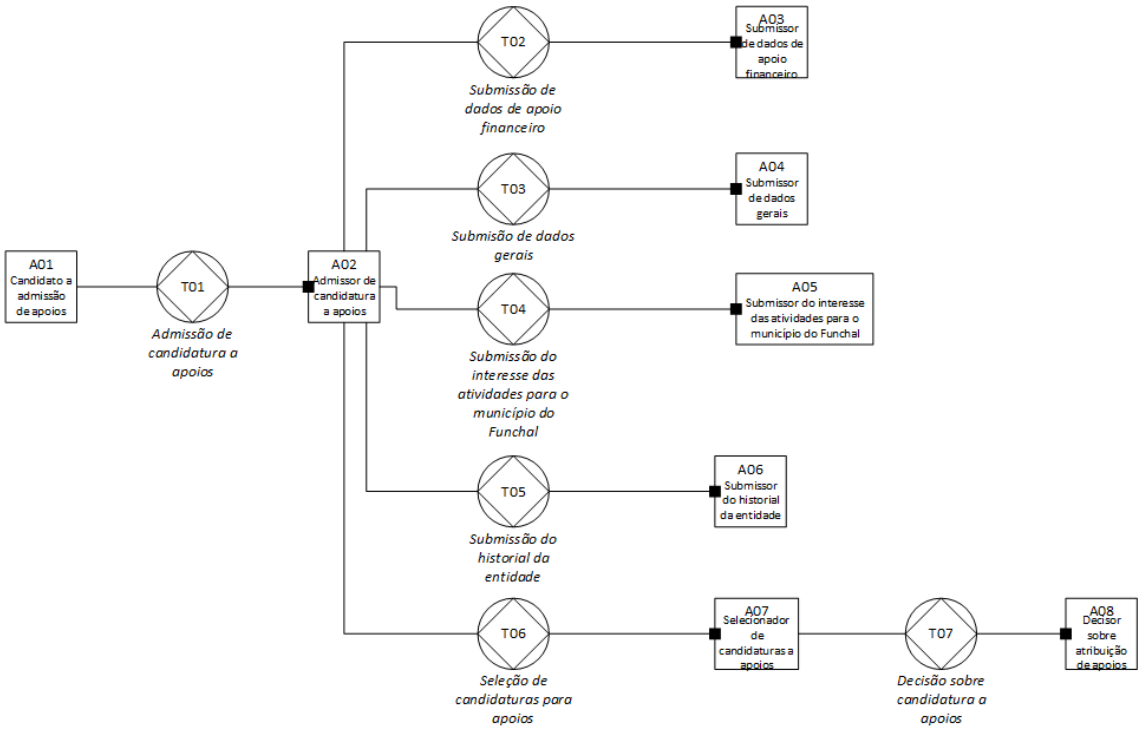


Figura 13 – ATD caso Apoios para o desenvolvimento de atividades de interesse municipal

3.2.4 PSD

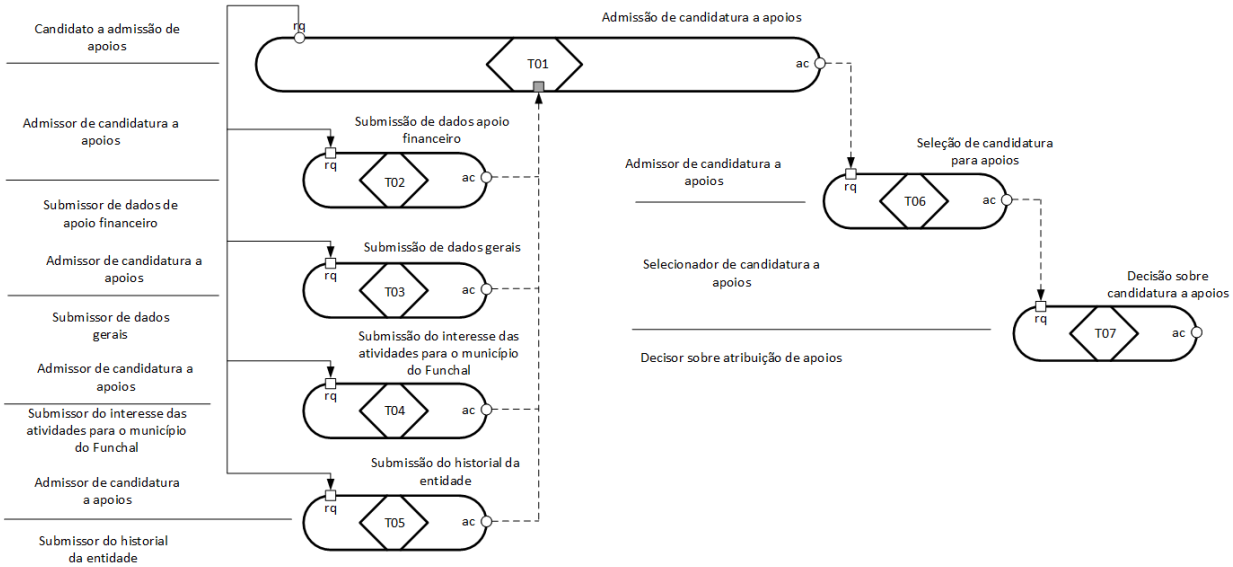


Figura 14 - PSD caso Apoios para o desenvolvimento de atividades de interesse municipal

3.2.5 OFD

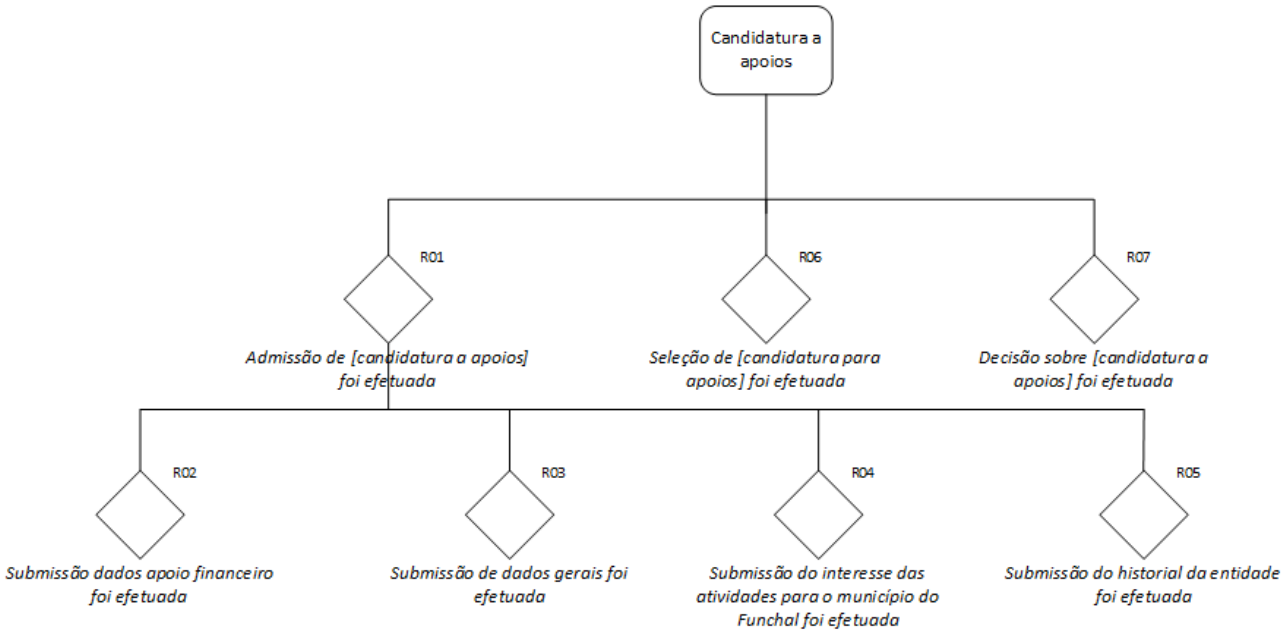


Figura 15 - OFD caso Apoios para o desenvolvimento de atividades de interesse municipal

4 Comparação de *Frameworks*

De modo a agilizar o processo de desenvolvimento, a utilização de uma *framework* é fundamental. Neste capítulo, primeiramente é esclarecido o conceito de *framework* e logo de seguida são apresentadas as *frameworks* PHP e Node.js analisadas. Para cada framework apresenta-se uma breve descrição e são expostas algumas vantagens e desvantagens. Posteriormente, são selecionadas *frameworks* para testes, e após os testes é finalmente decidido qual é a melhor *framework* para o desenvolvimento deste projeto.

4.1 Conceito de *framework*

Apesar do termo *framework* não possuir uma definição universal, podemos considerar que uma *framework* pode ser definida como uma estrutura formada por um conjunto de classes já implementadas e testadas, prontas para uso, em que um outro projeto de software pode ser organizado e desenvolvido. Uma *framework* é de extrema importância no desenvolvimento de software devido à organização e reutilização de código e tem como objetivo facilitar a vida do programador, ao tornar o desenvolvimento mais fácil, rápido, escalável e seguro [6].

4.2 Análise de *Frameworks*

Para apurar qual seria a melhor *framework* para desenvolver este projeto foram analisadas *frameworks* PHP e *frameworks* Node.js. Por sugestão do orientador, as *frameworks* PHP analisadas foram o CodeIgniter e o Laravel, e as frameworks Node.js foram Express e Meteor.

4.2.1 *Frameworks* Node.js

Antes de prosseguir com o estudo das *frameworks* é importante esclarecer alguns conceitos sobre o Node.js.

Node.js é um ambiente de execução JavaScript *open-source*, utilizado para executar código JavaScript no lado do servidor. Foi criado por Ryan Dahl em 2009 e a sua construção foi baseada no motor V8 JavaScript do Google, que é um motor *open-source* implementado pelo *Google* em C++ e utilizado no *Google Chrome* [7] [8].

Node.js utiliza um modelo de I/O (*Input/Output*) direcionado a eventos não bloqueante que o torna leve e eficiente, ideal para aplicações em tempo real com troca intensa de dados através de dispositivos distribuídos [9].

Para além disso, o Node.js utiliza o NPM (*Node Package Manager*) [10] como gestor de pacotes, que permite instalar, atualizar e desinstalar bibliotecas de uma forma simples [11].

4.2.1.1 Express

Express é uma *framework* minimalista e flexível para Node.js. É *open-source*, escrito em JavaScript e tem como objetivo facilitar a criação de aplicações web e móveis bem como APIs (*Application Programming Interface*), ao fornecer vários recursos úteis e poderosos [12] [13].

Vantagens [14] [15]:

- Fácil de configurar e customizar;
- Fácil de se conectar com bases de dados como MongoDB, Redis e MySQL;
- Promove a reutilização de código;
- Possui uma grande comunidade com vastos guias e tutoriais.

Desvantagens [14]:

- Exige tarefas manuais intensivas;
- Não oferece uma maneira universal de organizar coisas.

4.2.1.2 Meteor

Meteor é uma *framework open-source* para Node.js que utiliza a arquitetura MVC (*Model View Controller*) para a construção de aplicações web e móveis em tempo real. Para isso utiliza o protocolo DDP (*Distributed Data Protocol*) para propagar automaticamente as mudanças de dados para os clientes sem exigir que o desenvolvedor escreva qualquer código de sincronização.

Meteor integra-se com a base de dados noSQL MongoDB, e uma vez que é construído sobre Node.js, o Meteor utiliza JavaScript tanto no lado do cliente como no lado do servidor [16].

Vantagens [17]:

- Dinamiza o código e acelera o processo de desenvolvimento;
- Possui diversas bibliotecas e pacotes;
- É rápido.

Desvantagens [18]:

- Não há suporte para bases de dados SQL (*Structured Query Language*);
- Não é muito adequado para aplicações grandes e complexas;
- É necessário um conhecimento aprofundado do Node.js para personalizar completamente as aplicações.

4.2.2 *Frameworks* PHP

Por mais de uma década, a linguagem PHP continua a estar na lista das mais populares. Devido à alta taxa de adoção e de popularidade surgiram inúmeras *frameworks* PHP que possibilitam aos programadores desenvolver aplicações web mais complexas e seguras.

4.2.2.1 CodeIgniter

CodeIgniter é uma framework PHP, *open-source* para desenvolvimento de aplicações *web* que utiliza a arquitetura MVC. O objetivo do CodeIgniter é possibilitar que o utilizador desenvolva projetos mais rapidamente, através de um conjunto de bibliotecas direcionadas às tarefas mais comuns [19].

Vantagens [20]:

- Fácil de instalar e configurar;
- Baixa curva de aprendizagem;
- Documentação ampla e bem organizada;
- Bom desempenho.

Desvantagens [20]:

- Em comparação com outras *frameworks*, possui menos ferramentas e bibliotecas integradas;
- A própria framework não possui ORM (*Object-Relational Mapping*) integrado.

4.2.2.2 Laravel

Laravel é uma *framework* PHP criado por Taylor Otwell em 2011 para o desenvolvimento de aplicações *web*, é *open-source* e utiliza a arquitetura MVC. O seu principal objetivo é ajudar a desenvolver aplicações seguras de forma rápida, com código simples e limpo. Também possui várias características interessantes, tais como um sistema modular com um gestor de dependências dedicado, várias formas de acesso a bases de dados relacionais, a sua sintaxe é simples e concisa, e ainda possui várias ferramentas indispensáveis no auxílio ao desenvolvimento e manutenção de sistemas [21].

Vantagens [22]:

- Facilidade de uso;
- É baseado na arquitetura MVC;
- Tem uma grande comunidade a fornecer suporte;
- Documentação completa e de fácil compreensão;
- Baixa curva de aprendizagem;
- Aumento da produtividade e reaproveitamento de código.

Desvantagens [22]:

- Complexidade da estrutura dificulta a instalação em geral;
- Maior quantidade de conteúdo para aprender devido à grande quantidade de recursos.

4.3 Seleção de *Frameworks* para testes

Através da pesquisa efetuada, e tendo em conta as características, vantagens e desvantagens, numa fase inicial foi possível apurar quais seriam as duas *frameworks* mais adequadas para o desenvolvimento deste projeto.

Devido à natureza muito relacional do nosso esquema, a *framework* Meteor foi excluída visto que não possui suporte para bases de dados SQL**.**

O CodeIgniter também foi excluído das opções, porque comparando com o Laravel, não é tão robusto e possui menos bibliotecas e ferramentas. Por exemplo [23]:

- O Laravel possui um sistema de migração que oferece a facilidade de alterar a estrutura da base de dados;

- O Laravel oferece uma ferramenta incorporada para a linha de comandos conhecida como *Artisan*, que permite realizar muitas tarefas que os desenvolvedores evitam realizar manualmente;
- O Laravel oferece o *Eloquent ORM* que permite aos desenvolvedores fazer consultas à base de dados com a sintaxe PHP, em vez de escrever código SQL.

Estes são apenas alguns exemplos de recursos que se podem revelar uma mais valia no desenvolvimento de uma aplicação, e como o CodeIgniter não dispõe dos recursos que o Laravel oferece foi decidido excluir a framework CodeIgniter das opções.

Restando apenas o Express e o Laravel, para determinar qual destes seria o mais indicado para desenvolver este projeto foi realizado um pequeno teste com cada um, que será apresentado logo de seguida na secção 4.4.

4.4 Testes com Frameworks

Dispondo apenas das *frameworks* Express e Laravel, foi efetuado um pequeno teste com ambas, em que o objetivo era implementar as quatro operações básicas em bases de dados relacionais: criar, consultar, editar e remover dados.

Para ambos os testes, foi utilizada uma base de dados muito simples com o nome "testes" que continha apenas a tabela cliente com os campos "id", "nome" e "morada".

4.4.1 Express

Para começar a realizar o teste com o Express foi necessário instalar o Node.js. Posteriormente, foi criada uma pasta para conter o teste e foi utilizado o comando *npm init* para criar um ficheiro *package.json* para a nova aplicação. *Package.json* é o ficheiro responsável por descrever a aplicação, e contém dados como a versão do Node e do *npm*, dados do autor, dependências do projeto, entre outras coisas [24].

Seguidamente passou-se à instalação do Express através do comando *npm install express –save*, e obteve-se a estrutura de pastas que está representada na figura 16.

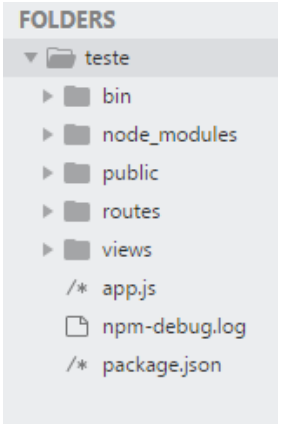


Figura 16 - Estrutura de pastas da aplicação criada com Express

A estrutura obtida é apenas uma das muitas maneiras de estruturar aplicações Express. O desenvolvedor pode utilizar esta estrutura ou modificá-la para melhor atender às suas necessidades.

De seguida é apresentado uma pequena descrição das pastas presentes na estrutura criada [25]:

- **bin**: esta pasta contém apenas um único ficheiro (*www*) que serve para ativar o servidor Node;
- **node_modules**: é a pasta onde todos os pacotes *npm* irão residir;
- **public**: contém todos os recursos necessários para a aplicação que não estão relacionados com o código Node.js. Esta pasta inclui ficheiros e pastas públicas, como imagens, JavaScript, CSS (*Cascading Style Sheets*), etc;
- **routes**: é a pasta onde devem ser definidas todas as rotas para a aplicação;
- **views**: é a pasta que é responsável por exibir dados para o utilizador através das vistas.

O ficheiro *app.js* é a base do projeto Node.js. Quando um utilizador acede à aplicação e faz um pedido, o ficheiro *app.js* vai analisar o pedido e descobrirá como lidar com a resposta dependendo do URL.

O ficheiro *app.js* utilizado está representado na figura 17.

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var cookieParser = require('cookie-parser');
var bodyParser = require('body-parser');
var index = require('./routes/index');
var users = require('./routes/users');
var test = require('./routes/test');
var app = express();

app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
app.use('/', index);
app.use('/users', users);
app.use('/test', test);
app.use(express.static('public'))

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

Figura 17 - Ficheiro app.js

No ficheiro *app.js* foram realizadas poucas alterações; apenas foram adicionadas as linhas 9 e 21 da figura 17. Na linha 9 podemos observar a função *require* que é uma função Node que importa um objeto de outro ficheiro ou módulo. Já na linha 21 é utilizado o método *app.use* que informa a aplicação para utilizar os parâmetros passados; neste caso o primeiro parâmetro é o caminho, e o segundo é a função a executar.

Posteriormente, dentro da pasta *routes* foi criado um ficheiro "test.js" onde foi efetuada a conexão à base de dados que está representada na figura 18.

```
//Conexão com a base de dados
var mysql = require('mysql')
var connection = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '',
  database : 'testes'
});
```

Figura 18 - Conexão à base de dados utilizando Express

Também foi no ficheiro "test.js", apresentado na figura 19, que a lógica e todas as rotas do sistema foram definidas.

```
15  router.get('/', function(req, res, next) {
16      res.sendFile('/teste/views/vista.html');
17  });
18
19  router.get('/user', function(req, res, next) {
20      connection.query('SELECT * FROM cliente LIMIT 0,15 ', function (err, rows, fields) {
21          if (err) throw err
22          res.send(rows)
23      })
24  });
25
26  router.delete('/user', function(req, res, next) {
27      var dados = req.body;
28      var sql = "DELETE FROM cliente WHERE id = '" + dados['id'] + "'";
29      connection.query(sql, function (err, result) {
30          res.send(result);
31      })
32  });
33
34  router.put('/userdados', function(req, res, next) {
35      var dados = req.body;
36      console.log(dados);
37      connection.query("SELECT * FROM cliente WHERE id = '"+ dados['id'] +"' ", function (err, rows, fields) {
38          if (err) throw err
39          res.send(rows)
40      })
41  });
42
43  router.put('/user', function(req, res, next) {
44      var dados1 = req.body;
45      var sql = "UPDATE cliente SET nome = '"+dados1['name']+"', morada = '"+dados1['morada']+"'  WHERE id = '" + dados1['id'] + "'";
46      connection.query(sql, function (err, result) {
47          res.send(result);
48      })
49  });
50
51  router.post('/user', function(req, res, next) {
52      var dados2 = req.body;
53      var sql = "INSERT INTO cliente (id, nome, morada) VALUES ('NULL', '"+dados2['nomeUser']+"', '"+dados2['moradaUser']+"') ";
54      connection.query(sql, function (err, result) {
55          res.send(result);
56      })
57  });
58
59  module.exports = router;
```

Figura 19 – Definição de rotas e lógica do sistema

Em termos de apresentação, foi criada a vista *vista.html* que continha um formulário que incluía apenas dois campos (nome e morada) e ainda uma tabela que apresentava os dados dos utilizadores presentes na base de dados. A vista está representada na figura 20.

```
<body>
  <h3>Nome</h3>
  <input type="hidden" id="idUser" name="idUser" value="">
  <input id="user-name" type="text" />
  <h3>Morada</h3>
  <input id="user-morada" type="text" />
  <input id="user-submit" type="button" value= "Inserir" onclick="inserir()" />
  <input id="user-cancelar" type="button" value= "Cancelar" style="display: none;" onclick="cancelar()" />
  <p id="output"></p>
  <table id="table" border="2px">
      <tr>
          <th>Nome</th>
          <th>Morada</th>
          <th>Remover</th>
          <th>Editar</th>
      </tr>
  </table>
</body>
```

Figura 20 – Vista vista.html

Após a vista estar completamente carregada, passou-se ao preenchimento da tabela com os dados provenientes da base de dados.

Como podemos observar na figura 21, é feito um pedido *ajax,* em que o *url é "test/user"* e o *type é "GET*". É através destes parâmetros que sabemos que rotas serão utilizadas.

```
$(document).ready(function () {
    $.ajax({
        url: "test/user",
        type: "GET",
        contentType: "application/json",
        processData: false,
        complete: function (data) {
            var json1 = JSON.parse(data.responseText);
            if (json1.length != 0) {
                $.each(json1, function(i, item) {
                    $("#table").append("<tr><td>"+ item.nome +"</td><td>"+ item.morada +"</td><td><a href='#' onClick='remover("+ item.id +")'>Remover</a></td>
                        <td><a href='#' onClick='editar("+ item.id +")'>Editar</a></td></tr>");
                });
            } else {
                $("#table").append("<tr colspan='3'><td>Não tem clientes..</td></tr>");
            }
        }
    });
});
```

Figura 21 - Preenchimento da tabela com os dados provenientes da base de dados

Os restantes métodos implementados são apresentados nas figuras 22 e 23.

```
function remover(idUser) {
    var dados = {id: idUser};
    $.ajax({
        url: "test/user",
        type: "DELETE",
        contentType: "application/json",
        processData: false,
        data: JSON.stringify(dados),
        complete: function (resultado) {
            if (resultado.responseText == "") {
                alert("Ocorreu um erro...");
            } else {
                location.reload();
            }
        }
    });
}

function editar(idUser) {

    var dados1 = {id: idUser};
    $("#idUser").val(idUser);
    $("#user-cancelar").show();
    $.ajax({
        url: "test/userdados",
        type: "PUT",
        contentType: "application/json",
        processData: false,
        data: JSON.stringify(dados1),
        complete: function (resultado) {
            console.log(resultado);
            var json1 = JSON.parse(resultado.responseText);
            $('#user-name').val(json1[0].nome);
            $('#user-morada').val(json1[0].morada);
            console.log(json1);
        }
    });
}

function cancelar() {
    $("#idUser").val("");
    $("#user-cancelar").hide();
}
```

Figura 22 - Métodos remover, editar e cancelar

```
function inserir() {
    var idUser = $("#idUser").val();
    var novoNome = $("#user-name").val();
    var novaMorada = $("#user-morada").val();
    if (idUser == "") {
        var dados2 = {nomeUser: novoNome, moradaUser: novaMorada};
        $.ajax({
            url: "test/user",
            type: "POST",
            contentType: "application/json",
            processData: false,
            data: JSON.stringify(dados2),
            complete: function (resultado) {
                if (resultado.responseText == "") {
                    alert("Ocorreu um erro...");
                } else {
                    location.reload();
                }
            }
        });
    } else {
        var dados1 = {id: idUser, name: novoNome, morada: novaMorada};
        $.ajax({
            url: "test/user",
            type: "PUT",
            contentType: "application/json",
            processData: false,
            data: JSON.stringify(dados1),
            complete: function (resultado) {
                if (resultado.responseText == "") {
                    alert("Ocorreu um erro...");
                } else {
                    location.reload();
                }
            }
        });
    }
}
```

Figura 23 - Método inserir

4.4.2 Laravel

Antes de instalar o Laravel é fundamental atender a alguns requisitos mínimos, particularmente possuir o *xampp* e o *composer* instalados. O *composer* [26] é uma ferramenta para gestão de dependências PHP, e visto que o Laravel utiliza o *composer* para gerir as suas dependências, o *composer* teve de ser previamente instalado. Foi instalada a versão 4.5 do composer e posteriormente a versão 5.4 do Laravel através do comando *composer create-project laravel/laravel –prefer-dist* e obteve-se a estrutura de pastas que está representada na figura 24.

Figura 24 Estrutura de pastas da aplicação criada com Laravel

Apesar desta ser a estrutura padrão, o *Laravel* não impõe quaisquer restrições de organização permitindo assim ao desenvolvedor organizar a sua aplicação conforme as suas preferências.

De seguida é apresentado uma pequena descrição das pastas presentes na estrutura padrão [27]:

- **app**: é uma das pastas mais importantes da estrutura pois contém o código principal da aplicação;
- **bootstrap**: contém ficheiros de configuração do *autoloading*;
- **config**: como o próprio nome indica contém todos os ficheiros de configuração da aplicação;
- **database**: contém os ficheiros de migração e os *seeds* da base de dados;
- **public**: contém o ficheiro *index.php* que é o controlador frontal, responsável pelo processo de inicialização e que encaminha todos os pedidos adequadamente. É também nesta pasta que todos os ficheiros públicos como imagens, CSS e JavaScript estão localizados;
- **resources**: e contém as vistas, os ficheiros de idiomas ficheiros LESS *(Leaner Style Sheets)* ou SASS *(Syntactically Awesome StyleSheets)*;
- **routes**: contém todas as definições de rotas para a aplicação;
- **storage**: é onde os ficheiros de *caches*, ficheiros *logs* e os ficheiros do sistema compilado estão localizados;

- **tests**: contém os testes automatizados;
- **vendor**: é onde o *composer* instala as suas dependências.

O ficheiro .env contém as variáveis de ambiente (variáveis que se espera que sejam diferentes em cada ambiente, e que, portanto, não são comprometidas com o controlo de versão).

Para fazer a conexão à base de dados apenas foi indicado no ficheiro *.env* o *username* e o nome da base de dados. A conexão à base de dados encontra-se representada na figura 25.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=testes
DB_USERNAME=root
DB_PASSWORD=
```

Figura 25 - Conexão à base de dados no Laravel

No Laravel, através dos ficheiros de migração, é possível criar e manipular a base de dados de uma forma simplificada e estruturada. Os ficheiros de migração são ficheiros que contêm uma classe com dois métodos: *up* e *down*.

O método *up* é usado para adicionar novas tabelas, colunas ou índices na base de dados enquanto que o método *down* tem a função de reverter as operações feitas pelo método *up* [28].

Para obter o ficheiro de migração e respetivo modelo, foi utilizado o comando *php artisan make:model Cliente –m.*

Quando a migração for executada, através do comando *php artisan migrate*, o método *up* irá criar uma tabela cliente com os campos: id, nome, morada, e duas colunas criadas pelo método *timestramps()* que são *created_at* e *updated_at*. O ficheiro de migração do cliente está representado na figura 26.

```php
class CreateClienteTable extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('cliente', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nome');
            $table->string('morada');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('cliente');
    }
}
```

Figura 26 - Ficheiro de migração com os métodos up e down

Para cada tabela que se encontra na base de dados, existe um modelo correspondente através do qual o desenvolvedor interage com a referida tabela. No modelo, representado na figura 27, é necessário identificar o nome da tabela da base de dados a manipular (linha 9), se queremos que as colunas *created_at* e *updated_at* sejam geradas automaticamente (linha 11), e também é necessário especificar um atributo *fillable* ou *guarded* (linha 13 a 18), uma vez que quando inserimos informação na base de dados esta é feita através de *mass-assignment* (atribuição em massa).

A vulnerabilidade de *mass-assignment* ocorre quando um utilizador passa um parâmetro HTTP (*HyperText Transfer Protocol*) inesperado através de um pedido, e esse parâmetro altera uma coluna da base de dados imprevisivelmente. Por exemplo, um utilizador mal-intencionado pode enviar um parâmetro *is_admin* através de um pedido HTTP, que é passado para o método *create* do modelo, permitindo ao utilizador escalar-se para um administrador [29].

```php
<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

class Cliente extends Model
{
    protected $table = 'cliente';

    public $timestamps = true;

    protected $fillable = [
        'nome',
        'morada'
    ];

    protected $guarded = [];
}

```

Figura 27 - Modelo Cliente

Posteriormente foi criado o controlador para conter toda a lógica necessária através do comando *php artisan make:controller ClienteController*. Nesse controlador foram criados métodos para inserir, editar, remover e apresentar dados. O controlador efetuado encontra-se representado na figura 28.

```php
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Cliente;

class ClienteController extends Controller
{
    public function insert (Request $request) {

        $nome = $request->input('usr');
        $morada = $request->input('morada');
        $data = ['nome'    => $nome, 'morada' => $morada];
        Cliente::create($data);
        return redirect('/');
    }

    public function show () {

        $users = Cliente::all();
        return view('welcome', compact('users'));
    }

    public function remove($idUser) {

        Cliente::find($idUser)->delete();
        return redirect('/');
    }

    public function showUpdateForm($idUser) {

        $cliente = Cliente::find($idUser);
        return view('formUpdate', compact('cliente'));
    }

    public function update($idUser, Request $request)  {

        $nome = $request->input('usr');
        $morada = $request->input('morada');
        $data = [ 'nome' => $nome, 'morada' => $morada ];
        Cliente::where('id', $idUser)->update($data);
        return redirect('/');
    }
}
```

Figura 28 - Controlador ClienteController

Em termos de apresentação, foi criada a vista *welcome.blade.php* que continha um pequeno formulário, com os campos nome e morada e ainda uma tabela que listava os dados dos utilizadores presentes na base de dados. A vista *welcome.blade.php* está presente na figura 29.

```
<body>
    <div>
        <div class = "row">
            <div class="col-md-4">
                <div class = "form-group">
                    <form method = "POST" action = "insert" > {{csrf_field()}}
                        <div class = "form-group">
                            <label for="usr">Name:</label>
                            <input type="text" class="form-control" id="usr" name ="usr">
                        </div>
                        <div class = "form-group">
                            <label for="morada">Morada:</label>
                            <input type="text" class="form-control" id="morada" name="morada">
                        </div>
                        <div class = "form-group">
                            <button type = "submit" class = "btn btn-default"> Inserir </button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
        <div class = "row">
            <div class="col-md-4">
                <table class="table table-condensed table-bordered table-hover">
                    <thead>
                        <tr>
                            <th>Nome</th>
                            <th>Morada</th>
                            <th>Remover</th>
                            <th>Editar</th>
                        </tr>
                    </thead>
                    <tbody>
                        @foreach($users as $user)
                        <tr>
                            <td>{{$user->nome}}</td>
                            <td>{{$user->morada}}</td>
                            <td><a href = "remove/{{$user->id}}" >Remover</a></td>
                            <td><a href = "updateForm/{{$user->id}}">Editar</a></td>
                        </tr>
                        @endforeach
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</body>
```

Figura 29 - Vista welcome.blade.php

Também foi criada a vista *formUpdate.blade.php,* representada na figura 30, que continha o formulário para editar os dados do utilizador.

```
<body>
    <div>
        <div class = "row">
            <div class="col-md-4">
                <div class = "form-group">
                    <form method = "POST" action = "/update/{{$cliente->id}}" >
                        {{csrf_field()}}
                        <div class = "form-group">
                            <label for="usr">Name:</label>
                            <input type="text" class="form-control" id="usr" name ="usr" value="{{$cliente->nome}}">
                        </div>
                        <div class = "form-group">
                            <label for="morada">Morada:</label>
                            <input type="text" class="form-control" id="morada" name="morada" value="{{$cliente->morada}}">
                        </div>
                        <div class = "form-group">
                            <button type = "submit" class = "btn btn-default"> Editar </button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</body>
```

Figura 30 - Vista formUpdate.blade.php

Em relação às rotas estas foram definidas no ficheiro web.php que está localizado dentro da pasta *routes*. A principal função de um sistema de rotas é mapear os pedidos HTTP para as áreas do sistema responsáveis por tratá-las. As rotas definidas podem ser visualizadas na figura 31.

```
Route::get('/', 'ClienteController@show');
Route::post('/insert', 'ClienteController@insert');
Route::get('/remove/{id}', 'ClienteController@remove');
Route::get('/updateForm/{id}', 'ClienteController@showUpdateForm');
Route::post('/update/{id}', 'ClienteController@update');
```

Figura 31 - Rotas

Como resultado final, para ambos os testes, obtivemos a interface que está representada na figura 32.

Nome:

Morada:

Inserir

Nome	Morada	Remover	Editar
Maria	Machico	Remover	Editar
José	Funchal	Remover	Editar
Inês	Lisboa	Remover	Editar
Alexandre	Porto	Remover	Editar

Figura 32 – Interface desenvolvida nos testes

4.5 Escolha da Framework

Após a realização dos testes com ambas as *frameworks* podemos concluir que o Laravel é a framework mais adequada para a implementação deste projeto. Apesar da *framework* Express ser muito popular e promissora, as razões para esta escolha basearam-se nomeadamente por existir uma maior familiarização com a linguagem PHP, o que se traduziu numa curva de aprendizagem menor, e consequentemente o tempo de desenvolvimento foi mais reduzido com a *framework* Laravel.

Outro aspeto positivo acerca do Laravel é que apresenta uma excelente documentação e possui uma comunidade ativa que se dedica a melhorar a framework e a fornecer suporte às pessoas associadas ao desenvolvimento.

5 Desenvolvimento do Protótipo

Após definir a *framework* a utilizar foi possível avançar para a etapa de desenvolvimento do protótipo. Neste capítulo será apresentado uma visão geral do protótipo e será descrito todo o processo de desenvolvimento adotado, desde a definição dos requisitos funcionais e não funcionais, a arquitetura, as tecnologias utilizadas, a implementação dos componentes com exemplos de utilização e por fim pelos testes de usabilidade efetuados com utilizadores.

É de relembrar que, anteriormente ao desenvolvimento que será apresentado de seguida, foi efetuado outros conteúdos que podem ser consultados no anexo A e B.

5.1 Visão geral do protótipo

Os objetivos do protótipo concentram-se em dois aspetos principais: permitir a modelação dos diagramas por meio de componentes e permitir a sua execução através de um *dashboard.*

O *dashboard* é um dos elementos mais importantes do protótipo, e o seu principal objetivo é controlar o fluxo de tarefas de um processo de acordo com o que está especificado nos componentes de modelação, ou seja, o fluxo de tarefas é executado conforme o que foi definido na especificação dos processos da organização respeitando a teoria, conceitos e diagramas do DEMO. O *dashboard* desenvolvido tem a particularidade de ter o seu conteúdo totalmente dinâmico, sendo que, durante a sua execução, gera automaticamente formulários com base nas propriedades que estão definidas para os tipos de entidades.

São vários os componentes que compõem o protótipo, mais especificamente:

- **Gestão de Idiomas**: componente onde é possível gerir o idioma em que a organização opera. No decorrer deste capítulo será apresentada a versão em português.
- **Gestão de Utilizadores**: componente no qual é possível inserir utilizadores e gerir os utilizadores que se encontram registados no protótipo. Neste componente também é possível associar os utilizadores a papéis previamente inseridos.
- **Gestão de Papéis**: componente onde é possível gerir os papéis e atribuí-los a utilizadores e atores primeiramente registados. Um papel pode pertencer a vários atores e a vários utilizadores.
- **Gestão de Atores**: componente no qual ocorre a gestão dos atores que devem ser associados aos papéis mencionados anteriormente. Um ator pode estar associado a vários papéis.
- **Gestão de Processos:** componente onde a gestão dos tipos de processos característicos de uma determinada organização pode ser efetuada.
- **Gestão de Transações:** componente responsável por gerir tipos de transações (que estão associados a tipos de processos), estados da transação e formulários customizados. Os tipos de transações referem-se às transações que foram obtidas através da modelação com a metodologia DEMO. Os estados da transação referem-se a atos de coordenação e produção, ou seja, ao pedido, promessa, execução,

afirmação e aceitação. Já os formulários customizados têm a finalidade de reunir formulários de vários tipos de transações para serem apresentados e preenchidos simultaneamente por um utilizador. É de salientar que neste componente também é possível definir e visualizar os atores iniciadores para cada tipo de transação.

- **Gestão de Entidades:** componente para gerir tipos de entidades, que estão associadas a um determinado tipo de transação. Pode-se considerar que um tipo de entidade é um objeto ou conceito do qual queremos guardar informação.
- **Gestão de Relações:** componente encarregado de gerir tipos de relações, que estão associadas a um determinado tipo de transação. Um tipo de relação consiste numa ligação entre dois tipos de entidades, o que corresponde a uma ligação de muitos para muitos entre duas tabelas na base de dados.
- **Gestão de Propriedades:** componente responsável pela gestão de propriedades dos tipos de entidades e dos tipos de relações. Pode-se considerar que uma propriedade é um atributo próprio que caracteriza um tipo de entidade ou um tipo de relação e que estará presente num formulário.
- **Gestão de Unidades:** componente onde as unidades das propriedades podem ser geridas. As unidades correspondem às unidades de um determinado atributo, como por exemplo: m (metros), cm (centímetros), € (euros) e não são obrigatórias.
- **Valores Permitidos:** componente onde é possível gerir os valores permitidos para uma determinada propriedade.
- **Diagrama de Estrutura do Processo:** componente onde as ligações causais e ligações de espera são especificadas conforme o PSD obtido na modelação.
- **Gestão de Pesquisas:** componente onde é possível efetuar e guardar pesquisas. As pesquisas são realizadas num determinado tipo de entidade e servem para visualizar todos os valores introduzidos conforme as propriedades selecionadas, valores e operadores introduzidos nesse tipo de entidade. As pesquisas realizadas podem ser guardadas, e os seus resultados podem ser exportados para formato Excel.

É de relembrar que este protótipo está a ser desenvolvido juntamente com dois colegas de mestrado e um bolseiro de investigação. Os componentes que ficaram inteiramente à minha responsabilidade foram: o Gestão de Relações, o Gestão de Propriedades e o Gestão de Pesquisas. Já a implementação dos restantes componentes foi dividida pelos outros elementos da equipa.

De modo a ilustrar a estrutura conceptual da base de dados foi elaborado um OFD que está representado na figura 33. As classes pintadas de cinzento correspondem às tabelas da base de dados responsáveis por armazenar a especificação dos processos da organização, consoante os diagramas obtidos pela metodologia DEMO. Em oposição, as classes pintadas de branco simbolizam as instâncias introduzidas, em conformidade com os processos a acontecer na organização.

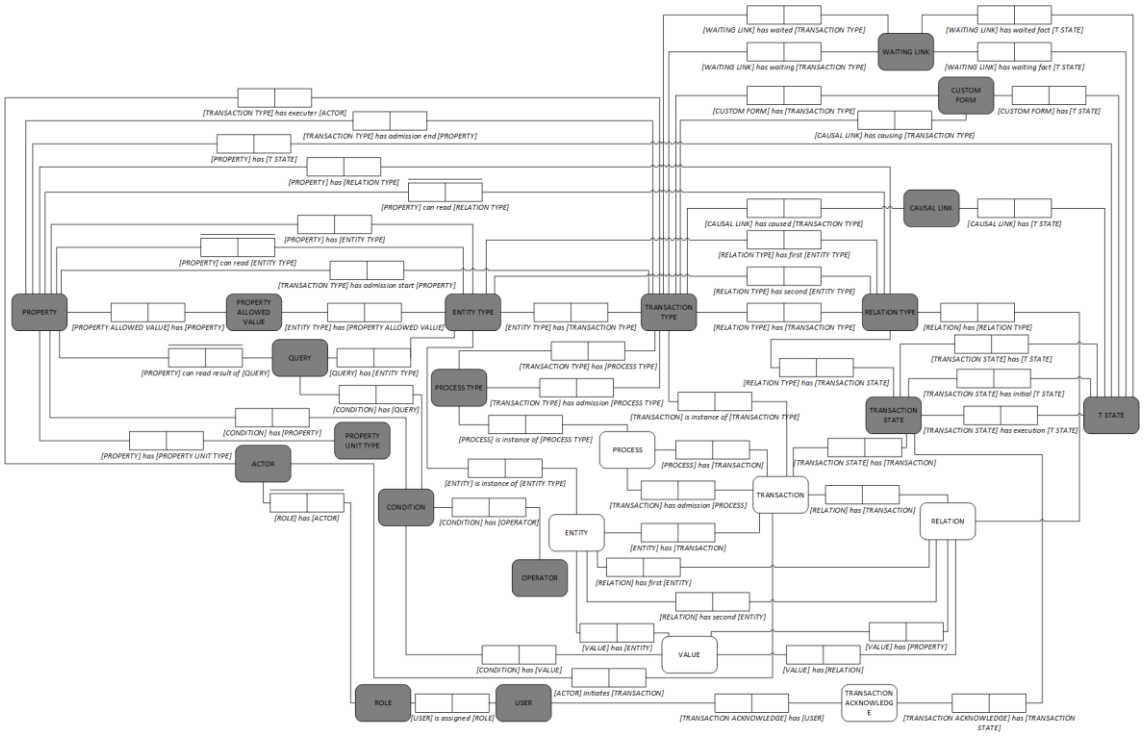


Figura 33 - OFD geral protótipo

Com o intuito de esclarecer sucintamente o diagrama exposto na figura 33, serão apresentadas várias figuras com partes mais específicas do referido diagrama.

O componente escolhido como ponto de partida será o tipo de processo pois refere-se às situações da organização que pretendemos modelar. Um tipo de processo (PROCESS TYPE) pode conter vários tipos de transações (TRANSACTION TYPE), e os tipos de transações estão sempre associados a um tipo de processo. Quer os tipos de processos quer os tipos de transações possuem instâncias (criadas quando ocorre a execução através do *dashboard*), que estão representadas pelas classes *PROCESS* e *TRANSACTION* respetivamente. Tal como um tipo de processo está associado a um tipo de transação, a instância de processo está identicamente associada a uma instância de transação.

A classe *TSTATE* refere-se aos atos de coordenação e produção de uma transação, por exemplo, ao pedido, promessa, execução, declaração e aceitação. No decorrer da execução, para cada instância de transação é importante saber em que estado se encontra, daí a existência da classe *TRANSACTION STATE*.

Já a classe *CUSTOM FORM* refere-se aos formulários customizados, que estão associados aos tipos de transações. Um formulário pode estar associado a vários tipos de transações e um tipo de transação pode estar associado a vários formulários.

Na figura 34 é apresentada uma visão mais específica das classes mencionadas anteriormente.

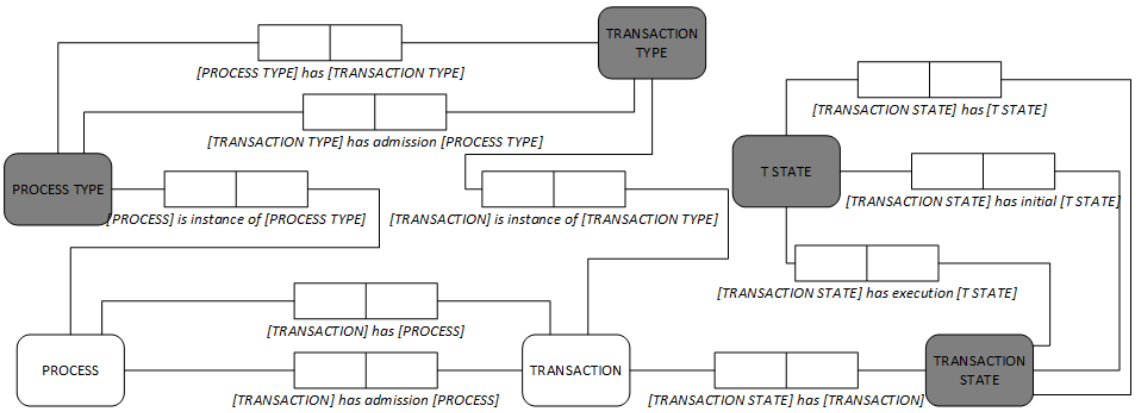


Figura 34 - OFD – tipos de transação e processo com suas instâncias, atos de coordenação e produção, formulários customizados

Ainda relativamente aos tipos de transações, estas precisam de ser iniciadas e executadas por alguém, por isso existem as classes *USER*, *ROLE* e *ACTOR*. Um utilizador pode desempenhar vários papéis, e esses papéis são desempenhados por atores, que podem ser iniciadores de uma transação e executores de um tipo de transação.

A classe *TRANSACTION ACKNOWLEDGE* tem apenas finalidade informativa, ou seja, serve para informar que o utilizador já viu a transação em particular. Na figura 35 estão representadas as classes referidas.

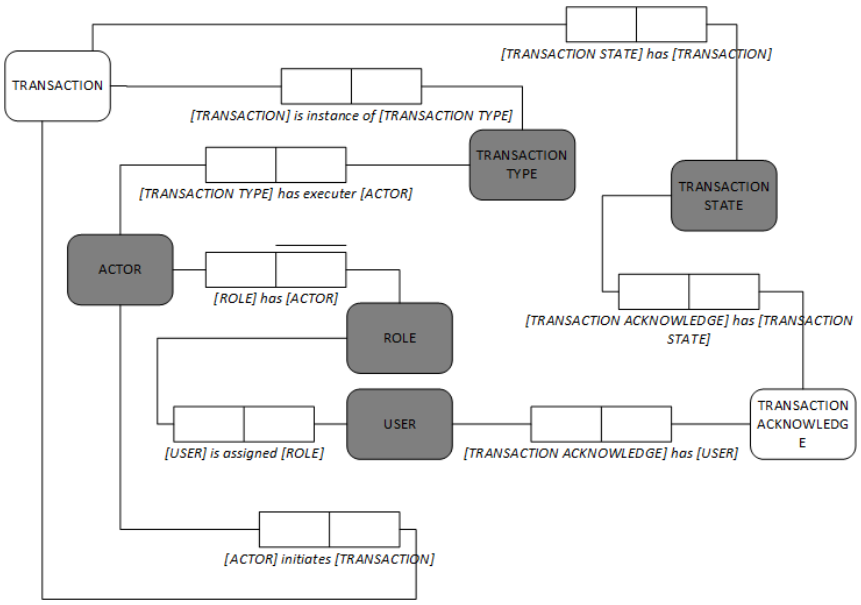


Figura 35 – OFD – Utilizadores, papéis e atores

No decorrer dos processos, as transações podem causar ou ter de esperar por outras. Desse modo é exigido especificar as ligações causais (*CAUSAL LINK*) e de espera (*WAITING LINK*). No que diz respeito às ligações causais, ou seja, quando um ato de coordenação de um tipo de transação provoca a ocorrência de um ato de coordenação ou produção de outro tipo de transação, é necessário referir qual é o tipo de transação causado, o tipo de transação que causa, respetivo ato de coordenação, e ainda a

multiplicidade. A multiplicidade refere-se ao número de vezes que a transação vai ser criada.

No que se refere às ligações de espera, isto é, quando um ato de coordenação ou produção de um tipo de transação tem de esperar pela ocorrência de um ato de coordenação de outro tipo de transação, é necessário especificar quais são esses tipos de transações, respetivos atos de coordenação ou de produção e a multiplicidade.

Na figura 36 é apresentada a parte do diagrama que corresponde ao que foi referido.

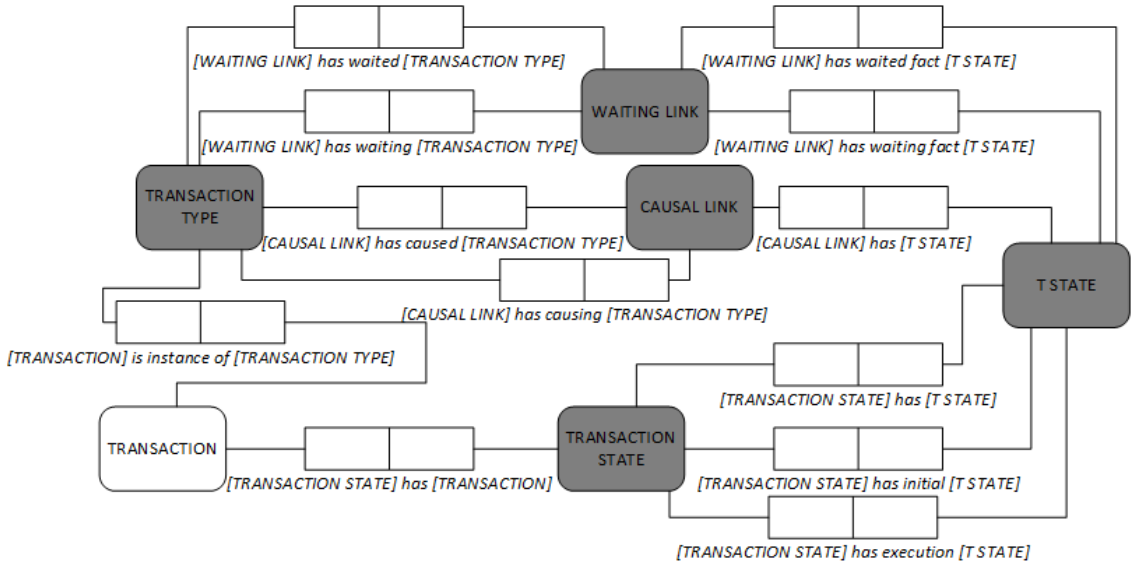


Figura 36- OFD - tipos de transações com ligações causais e de espera com respetivos atos

Como já tinha sido mencionado, um tipo de transação está sempre associado a um tipo de entidade (*ENTITY TYPE*) ou a um tipo de relação (*RELATION TYPE*). Os tipos de entidades e tipos de relações também possuem instâncias (*ENTITY* e *RELATION,* nessa ordem), que consequentemente estão associadas às instâncias de transações. Na figura 37 é apresentado uma parte do diagrama correspondente aos tipos de entidades.

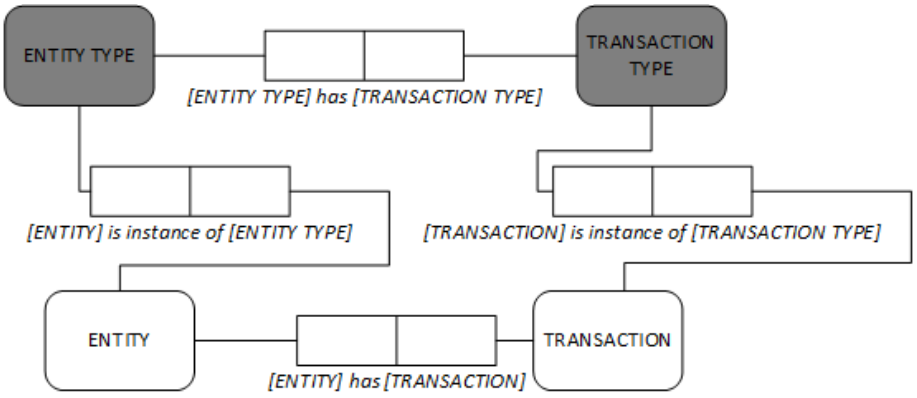


Figura 37 – OFD - tipo de transação associado a tipo de entidade e suas instâncias

Já na figura 38 é apresentado a parte do diagrama que corresponde aos tipos de relações. É de referir que um tipo de relação é constituído por dois tipos de entidades, portanto a instância de relação também é constituída por duas instâncias de entidades.

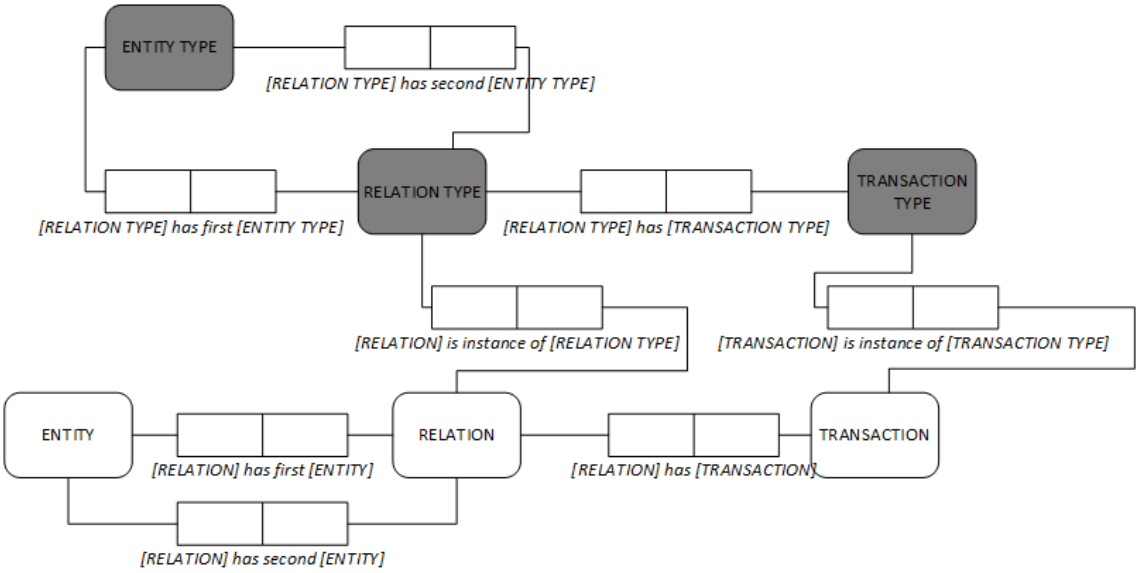


Figura 38 – OFD - tipo de transação associado a tipo de relações e suas instâncias

Quer os tipos de entidades quer os tipos de relações contêm propriedades (PROPERTY), que por sua vez podem ter unidades (*PROPERTY UNIT TYPE*) e valores permitidos (*PROPERTY ALLOWED VALUE*). Uma propriedade está sempre associada a um ato de coordenação ou de produção (TSTATE). Adicionalmente, na execução do *dashboard*, as propriedades conseguem ter acesso a dados de tipos de entidades, tipos de relações e até mesmo outras propriedades. A parte do diagrama que representa o que foi referido está presente na figura 39.

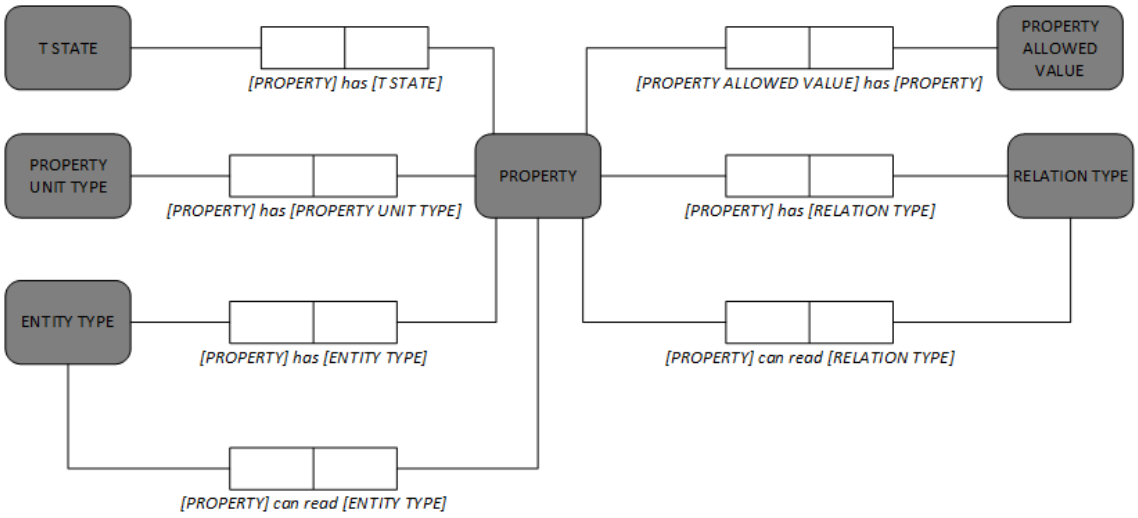


Figura 39 - OFD focando propriedades, unidades, valores permitidos

Para além disso existe a classe *VALUE* que corresponde ao valor de uma determinada propriedade em relação a uma certa instância de entidade ou instância de relação. Na figura 40 podemos encontrar uma parte do OFD a focar nesse aspeto.

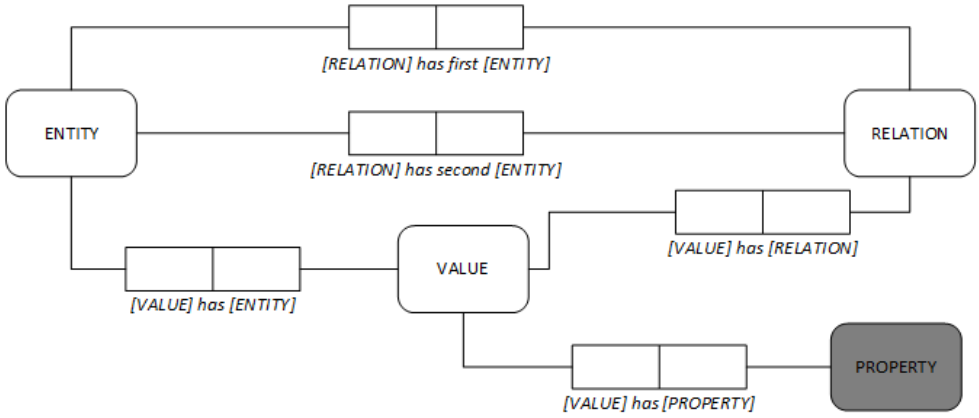


Figura 40 - OFD focando valores, instâncias e propriedades

Por último, falta apenas referir os componentes *QUERY*, *CONDITION* e *OPERATOR*. QUERY refere-se à *query* ou pesquisa a ser armazenada. Uma *query* é realizada num tipo de entidade e possui várias condições, mais concretamente quais as propriedades que foram selecionadas, e quais os operadores e valores inseridos. Na figura 41 está presente a parte do diagrama focada nesse aspeto.

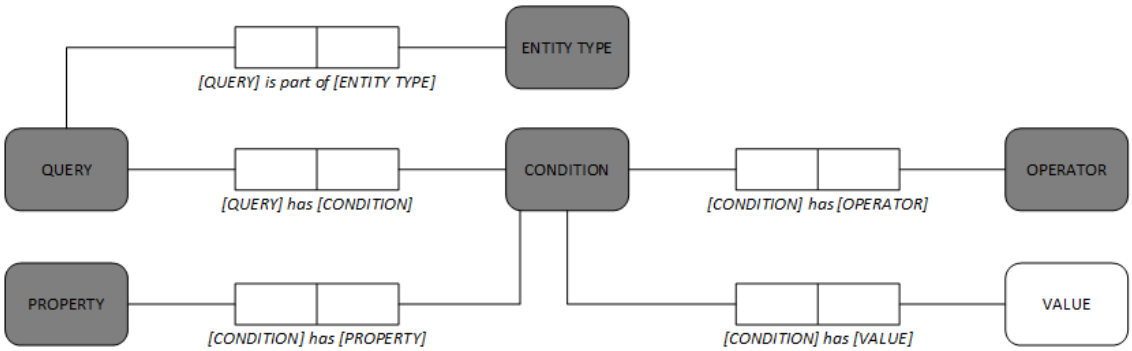


Figura 41 - OFD focando operadores, query e condições

5.2 Requisitos

Uma das etapas mais importantes no desenvolvimento de um sistema é a definição de requisitos. Os requisitos são capacidades, atributos ou características que o sistema deve necessariamente dispor para ser útil aos seus utilizadores [30]. Normalmente os requisitos são divididos em requisitos funcionais e não funcionais [31].

Como já tinha sido referido previamente, a implementação dos componentes foi dividida por uma equipa de quatro elementos, portanto os requisitos funcionais e não funcionais apresentados de seguida referem-se apenas aos componentes Gestão de Relações, Gestão de Propriedades e Gestão de Pesquisas, uma vez que ficaram sob minha responsabilidade.

5.2.1 Requisitos funcionais

Os requisitos funcionais, como o próprio nome sugere, dizem respeito às funcionalidades propriamente ditas de um sistema, ou seja, descrevem as diversas operações que os clientes e utilizadores desejam ou necessitam que o sistema ofereça

[31]. Na tabela 3 estão presentes os requisitos funcionais que foram identificados para cada componente a implementar.

Nº	***Requisito funcional***
RF01	O sistema deve permitir ao utilizador a criação de propriedades nos tipos de entidades.
RF02	O sistema deve permitir ao utilizador a criação de propriedades nos tipos de relações.
RF03	O sistema deve permitir ao utilizador a edição de propriedades.
RF04	O sistema deve permitir ao utilizador a visualização de propriedades.
RF05	O sistema deve permitir ao utilizador a reordenação de propriedades.
RF06	O sistema deve permitir ao utilizador a criação de tipos de relação.
RF07	O sistema deve permitir ao utilizador a edição de tipos de relação.
RF08	O sistema deve permitir ao utilizador a visualização de tipos de relação.
RF09	O sistema deve permitir ao utilizador selecionar um tipo de entidade para efetuar a pesquisa dinâmica.
RF10	O sistema deve permitir ao utilizador selecionar propriedades para efetuar a pesquisa dinâmica.
RF11	O sistema deve permitir ao utilizador inserir valores para efetuar a pesquisa dinâmica.
RF12	O sistema deve permitir ao utilizador selecionar operadores para efetuar a pesquisa dinâmica.
RF13	O sistema deve permitir ao utilizador exportar resultados de pesquisa para formato Excel.
RF14	O sistema deve permitir ao utilizador guardar pesquisas efetuadas.
RF15	O sistema deve permitir ao utilizador visualizar pesquisas guardadas.
RF16	O sistema deve permitir ao utilizador editar pesquisas guardadas.

Tabela 3 - Requisitos funcionais

5.2.2 Requisitos não funcionais

Os requisitos não funcionais, também denominados de atributos de qualidade, são aqueles que não estão diretamente relacionados à funcionalidade do sistema, mas sim a aspetos de qualidade que o sistema deve suportar. Estes requisitos podem dizer respeito a aspetos de segurança, usabilidade, desempenho, escalabilidade, manutenção, disponibilidade, confiabilidade, entre outros [32]. Os requisitos não funcionais identificados estão representados na tabela 4.

Nº	***Requisito não funcional***
RNF01	O sistema deve ser de fácil utilização para os seus utilizadores.
RNF02	A interface do sistema deve ser visualmente apelativa para os utilizadores.
RNF03	O sistema deve verificar campos preenchidos pelo utilizador de modo a evitar que o utilizador introduza dados incorretos.
RNF04	Em caso de introdução de dados incorretos, deve ser apresentado mensagens de erros informativos e orientados ao utilizador.
RNF05	O sistema deve funcionar corretamente em todos os browsers.
RNF06	O sistema deve ser *responsive*, ou seja, deve adaptar-se às características de diferentes dispositivos.

Tabela 4 - Requisitos não funcionais

5.3 Arquitetura do sistema

A arquitetura de um sistema de software consiste na estrutura do sistema que compreende os elementos de software, as relações entre estes elementos, e as propriedades externamente visíveis destes elementos. A definição da arquitetura é fundamental para obter uma visão geral do sistema, e compreender os elementos importantes do software [33].

Nas aplicações web a arquitetura é do tipo cliente-servidor. Nesse tipo de arquitetura o servidor fornece uma função ou um serviço para um ou mais clientes, que iniciam pedidos para tais serviços [34]. De uma forma geral, o utilizador acede ao sistema introduzindo o respetivo URL (*Uniform Resource Locator*) no *browser* (cliente), que por sua vez faz um pedido ao servidor através da internet utilizando os protocolos HTTP ou HTTPS *(Hyper Text Transfer Protocol Secure)*. Se for necessário, o servidor comunica com a base de dados, e posteriormente responde ao cliente com os dados solicitados, que podem ser em formato JSON (*JavaScript Object Notation*) ou HTML.

A arquitetura do sistema para este projeto encontra-se representada na figura 42.

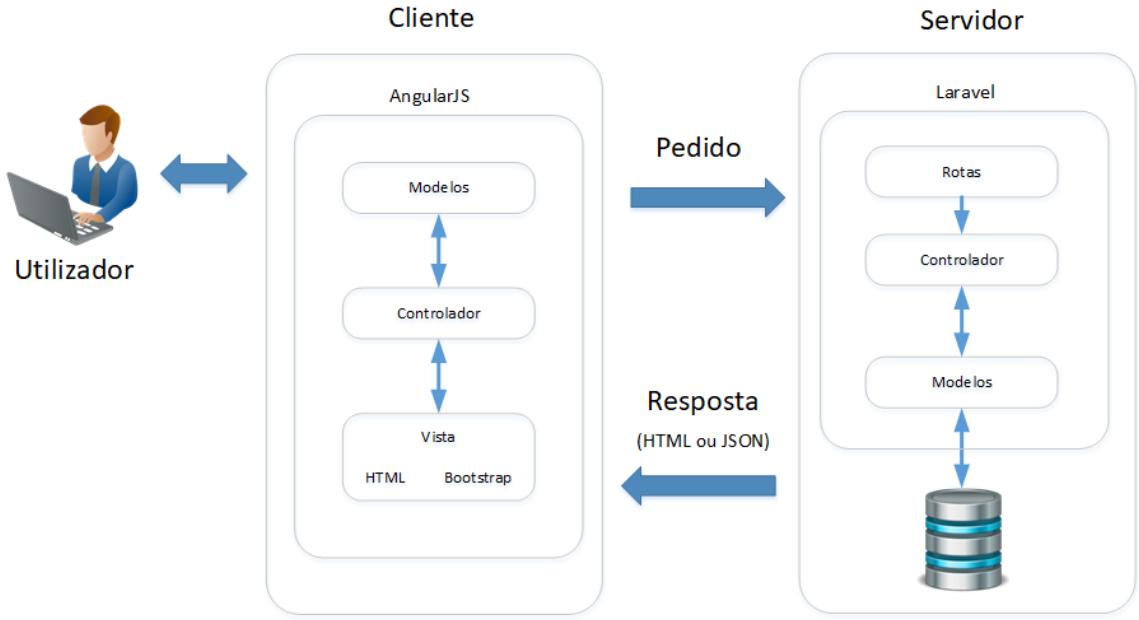


Figura 42 - Arquitetura do sistema

5.4 Tecnologias Utilizadas

Para além da *framework* Laravel, analisada no quarto capítulo, foram utilizadas outras tecnologias, nomeadamente o Bootstrap e o AngularJS que serão descritas de seguida:

5.4.1 AngularJS

Foi decidido utilizar a *framework* AngularJS, não só para tornar as páginas mais interativas para os utilizadores, mas também por ser simples de implementar com o Laravel. Além disso, a utilização de uma *framework* permite ter uma estrutura com o código mais organizado.

AngularJS é uma *framework* open-source para o desenvolvimento de aplicações web utilizando JavaScript. A *framework* AngularJS fornece um conjunto de bibliotecas que tornam o desenvolvimento de aplicações mais simples, e é utilizada principalmente em aplicações de página única (SPA – *Single Page Application*) [35]. Em aplicações de página única, todo o código necessário como HTML (*Hypertext Markup Language*), JavaScript e CSS, ou são obtidos com um único carregamento de página ou os recursos apropriados são carregados dinamicamente e adicionados à página conforme necessário, normalmente em resposta a ações do utilizador [36].

De seguida são apresentadas as características principais do AngularJS [37]:

- **Utiliza padrão MVC:** o AngularJS é construído sobre o conceito de MVC, permitindo assim uma melhor gestão da aplicação.
- **Utiliza recurso *Two-Way Data Binding*:** o AngularJS possibilita uma comunicação de dados bidirecional, ou seja, permite que qualquer alteração que ocorra nos modelos seja automaticamente refletida nas vistas e vice-versa.
- **Baseado no conceito de SPA:** o que proporciona ao utilizador uma aplicação dinâmica que apenas carrega recursos quando necessário.
- **Uso de diretivas:** as diretivas são extensões da linguagem HTML que oferecem a capacidade de estender o comportamento de elementos HTML. Este recurso permite a implementação de novos componentes de forma declarativa.
- **Utiliza o conceito de injeção de dependência:** a injeção de dependência é um padrão de design de software que lida com a forma como os componentes adotam as suas dependências. Desta forma o código torna-se mais organizado, flexível e testável.

5.4.2 Bootstrap

Um dos aspetos que tem mais impacto na experiência do utilizador nas aplicações web é o design. Para uma boa experiência a interface deve ser apelativa e também *responsive*, já que muitas pessoas acedem aos sites através de dispositivos móveis. Uma das *frameworks* mais conhecidas para criar aplicações web *responsive* é o Bootstrap.

O Bootstrap é uma *framework front-end open-source* que possui diversas classes CSS já prontas para uso e *plugins* em JavaScript (jQuery) que permitem resolver os problemas comuns no processo de desenvolvimento *front-end* [38].

Foi decidido optar pela utilização do Bootstrap neste projeto, não só por ser apelativo visualmente e para poupar tempo no desenvolvimento, mas também por conter as seguintes vantagens [39] [40]:

- **Reutilização de código**: o Bootstrap permite escrever menos código, uma vez que ele já dispõe uma série de formatações visuais prontas para uso. O desenvolvedor apenas tem que incluir o Bootstrap no projeto e saber que classes deve chamar.
- **Sistema baseado em *grid (grade)***: por ser baseado num sistema de *grid*, o Bootstrap permite a criação de sites *responsive* de forma fácil e intuitiva. O

sistema de *grid* utiliza uma série de *containers*, linhas e colunas para alinhamento do conteúdo e possibilita a divisão da tela em até doze colunas da mesma largura.

- **Fácil de utilizar**: o Bootstrap é fácil de utilizar e de compreender. Qualquer pessoa que tenha um conhecimento básico de HTML e CSS pode começar a utilizar o Bootstrap.
- **Ótimo suporte e documentação**: o Bootstrap possui uma comunidade muito grande e ativa, sempre dedicada a evoluir a *framework*, e para além disso, dispõe de uma documentação sempre atualizada e de fácil acesso.

5.5 Implementação dos componentes

A implementação dos componentes apresentada de seguida refere-se aos componentes pelos quais fiquei responsável: Gestão de Relações, Gestão de Propriedades e Gestão de Pesquisas. Para uma melhor compreensão dos componentes, serão exibidos vários *screenshots* da interface. Ao longo da explicação dos componentes também serão apresentados exemplos de utilização do protótipo para um melhor entendimento da implementação e das funcionalidades.

Entretanto será demonstrado como é que funciona o *dashboard*. Embora o *dashboard* não tenha ficado à minha responsabilidade, é importante conhecer o seu funcionamento de modo a compreender todo o protótipo.

5.5.1 Explicação do *dashboard*

Para efetuar um exemplo de utilização do *dashboard*, foram introduzidos todos os dados necessários. Ou seja, foram inseridos vários utilizadores, papéis de ator, atores, tipos de processos, tipos de transações, tipos de entidades, propriedades, valores permitidos, ligações causais, ligações de espera, etc. É de referir que todos estes dados foram inseridos através dos componentes em conformidade com os diagramas obtidos a partir da modelação com o DEMO, principalmente com o PSD da figura 14 definido anteriormente no terceiro capítulo “Cenários de Aplicação e Modelos”.

Para a demonstração deste exemplo será considerado o caso de Apoios para o desenvolvimento de atividades de interesse municipal. Para tal, irá assumir-se que vários utilizadores irão aceder ao sistema, mais concretamente desempenhando os papéis de ator de munícipe, membro da Câmara Municipal do Funchal, chefe de departamento e presidência.

Ao aceder ao protótipo, depois de efetuar o login, surge o *dashboard*, que é a página principal. O *dashboard* encontra-se dividido em quatro painéis diferentes:

- **Painel de tarefas do iniciador:** onde surgem os tipos de transações que o utilizador em sessão, caso seja iniciador, pode iniciar.
- **Painel de formulários personalizados:** em que aparecem os formulários customizados.
- **Painel das tarefas do iniciador:** onde estão presentes os tipos de transações que já foram iniciados pelo utilizador em sessão.

- **Painel de tarefas de execução:** no qual são exibidos os tipos de transações que o utilizador em sessão executou e que pode executar.

Para uma melhor compreensão, o *dashboard* do munícipe está representado na figura 43.

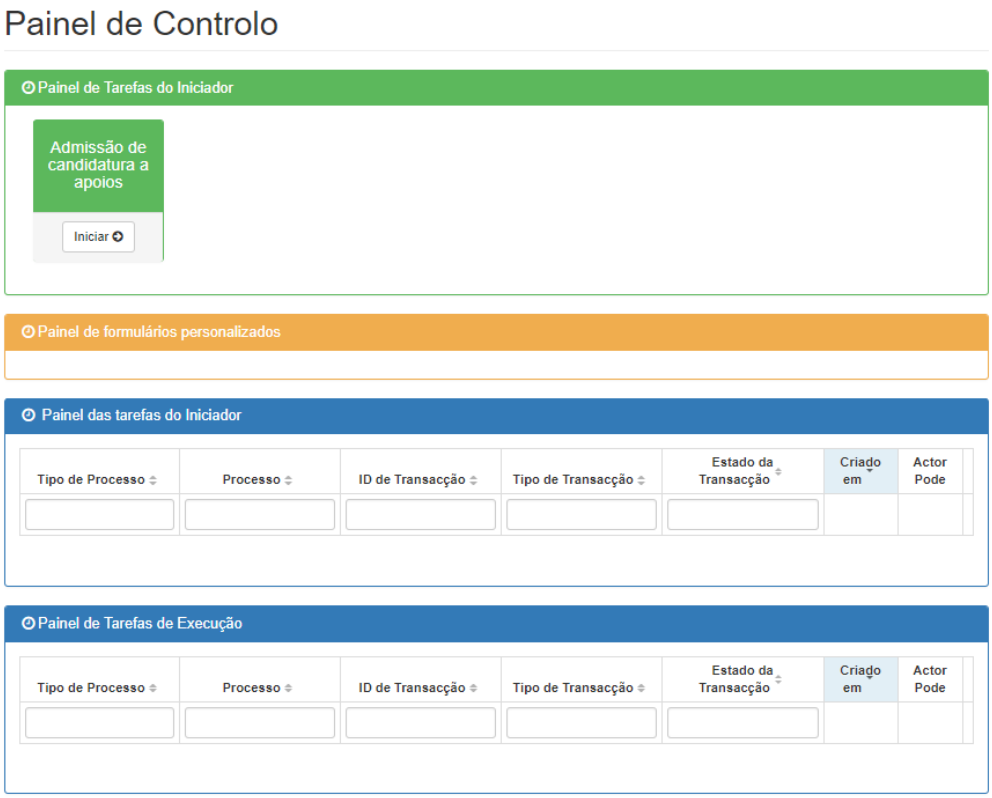


Figura 43 - Dashboard do munícipe

Como pode ser observado na figura 43, no primeiro painel aparece o tipo de transação que o munícipe pode iniciar. Até agora os restantes painéis encontram-se sem dados uma vez que ainda não foram introduzidos formulários personalizados, e porque, de momento, o utilizador não iniciou nem pode executar nenhum tipo de transação.

Neste exemplo, o objetivo do munícipe é realizar a admissão da sua candidatura a apoios para o desenvolvimento de atividades de interesse municipal. Para isso, apenas é necessário dirigir-se ao painel de tarefas do iniciador, que está representado na imagem 44, e clicar no botão “Iniciar”.

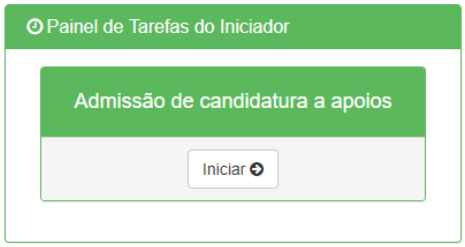


Figura 44 - Painel de tarefas do iniciador (munícipe)

Após clicar no botão, o pedido de admissão de candidatura a apoios é realizado. Segundo o PSD do caso, representado na figura 14, ao efetuar o pedido do tipo de transação "Admissão de candidatura a apoios" são desencadeados os quatro pedidos pertencentes aos tipos de transações "Submissão de dados gerais", "Submissão do interesse", "Submissão do historial da entidade" e "Submissão de dados apoio financeiro". Portanto, quando o pedido da admissão de candidatura é efetuado, automaticamente os pedidos desses quatro tipos de transações são realizados pelo sistema. Consequentemente, é possível visualizar uma nova versão do *dashboard* do munícipe, que se encontra na figura 45.

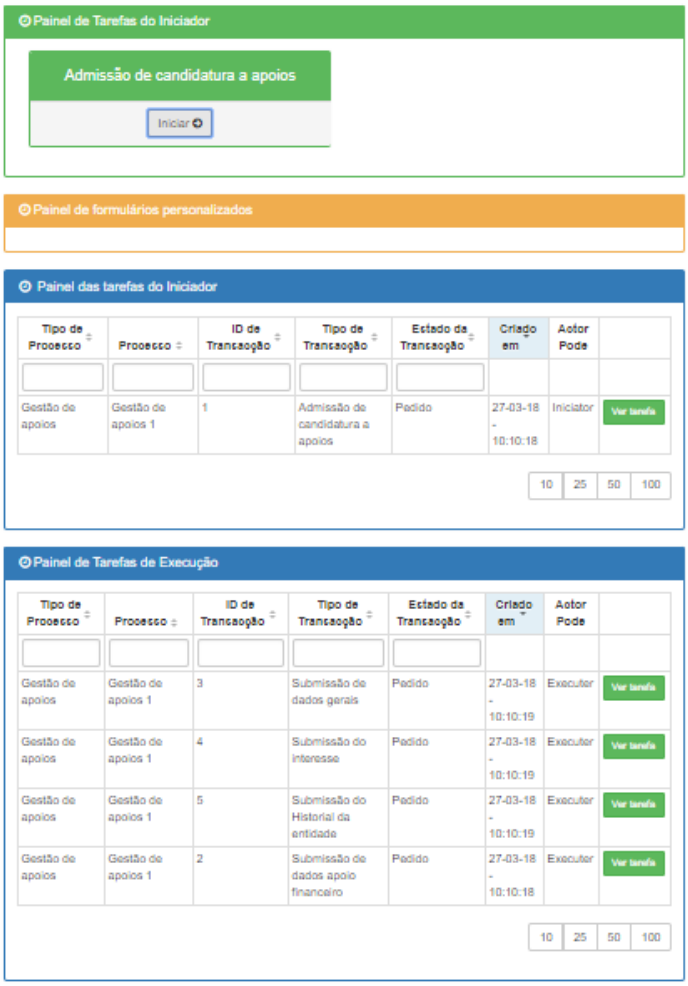


Figura 45 - Dashboard do munícipe após fazer o pedido de admissão de candidatura a apoios

Uma vez que o munícipe já efetuou o seu pedido, este fica registado no painel das tarefas do iniciador. Por outro lado, no painel de tarefas de execução, os tipos de transações que o munícipe pode executar já estão listados.

Ao clicar no botão "Ver tarefa" surge o *modal* que está na figura 46. Um *modal* é uma janela que aparece a partir de uma página principal. Neste *modal* o utilizador pode visualizar informação sobre o tipo de transação e o seu estado.

Figura 46 – Modal Informação da tarefa

Carregar no botão "Visto" é opcional e neste caso, serve apenas para informar ao iniciador do tipo de transação que o respetivo executor já a viu e é registado a data e a hora no campo "Visto da Transação". Posteriormente, é possível seguir para o próximo passo ao carregar no botão "Próximo passo da tarefa". Nesta ocasião, o munícipe já pode preencher os formulários de candidatura com os seus dados.

No caso do tipo de transação "Submissão de dados apoio financeiro" o formulário a preencher é o que está presente na figura 47.

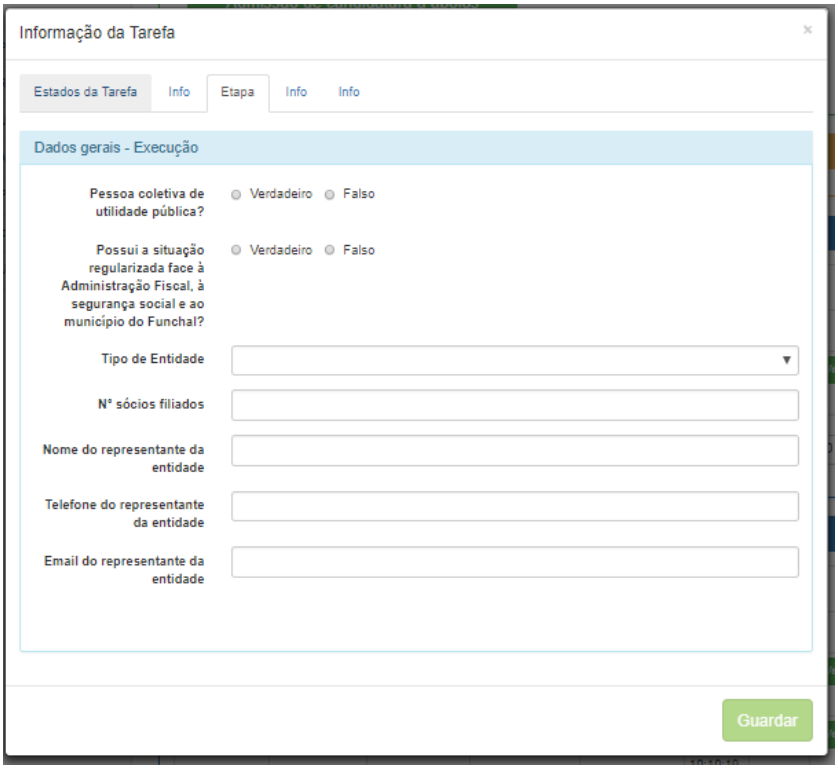


Figura 47 - Formulário "Submissão de dados apoio financeiro"

Após o preenchimento de todos os campos o botão "Guardar" desbloqueia e o munícipe pode guardar os seus dados. À medida que o estado da transação avança este é registado no painel de tarefas de execução, como pode ser visto na figura 48.

Figura 48 - Painel de tarefas de execução (estados da transação)

Procedendo para o tipo de transação "Submissão do interesse das atividades para o município do Funchal" o munícipe tem de preencher o que se encontra na figura 49.

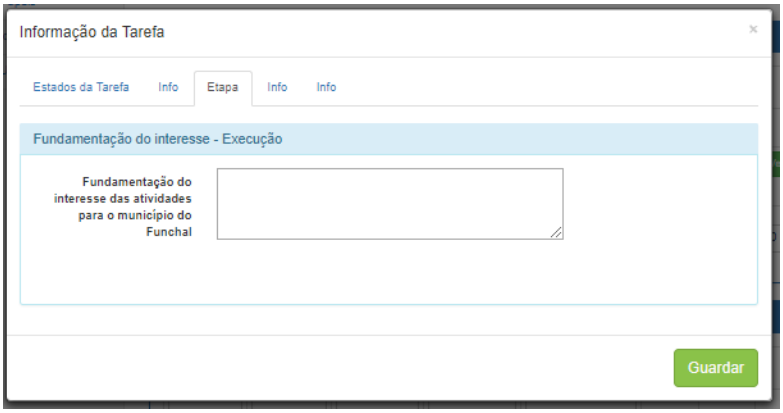


Figura 49 - Formulário "Submissão do interesse das atividades para o município do Funchal"

Em relação ao tipo de transação "Submissão do historial da entidade" surge o formulário que está representado na figura 50.

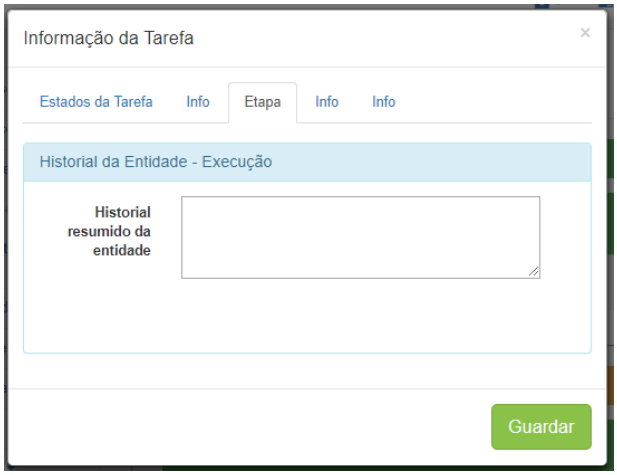


Figura 50 – Formulário "Submissão do Historial da Entidade"

E por fim, em relação ao tipo de transação "Submissão de dados apoio financeiro" surge o formulário que está na figura 51.

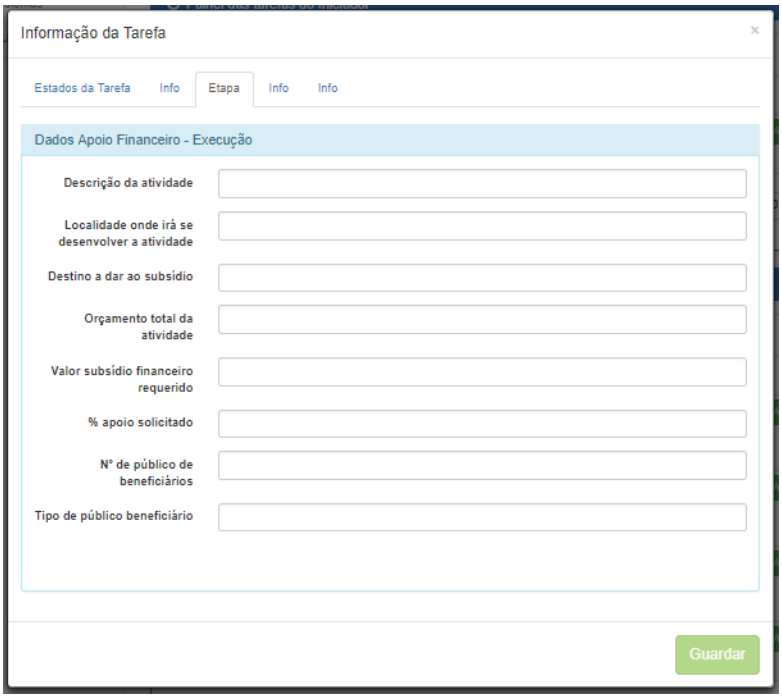


Figura 51 - Formulário "Submissão de dados apoio financeiro"

Após o munícipe ter submetido a sua candidatura compete ao *admissor de candidaturas a apoios* fazer uma pequena verificação dos dados. Caso as candidaturas sejam admitidas serão encaminhadas para o departamento adequado.

Assumindo que neste momento o utilizador em sessão desempenha o papel de ator *admissor de candidaturas a apoios*, quando este acede à plataforma, no seu painel de tarefas de execução, representado na figura 52, existe um tipo de transação para executar.

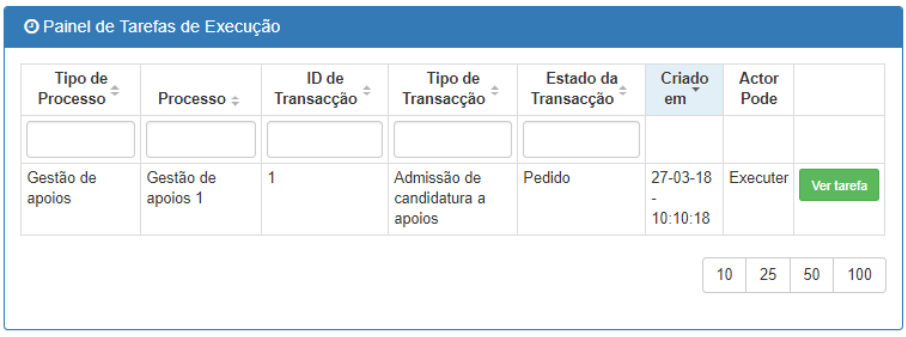


Figura 52 – Painel de tarefas de execução (admissor canidadturas)

Tal como anteriormente, ao clicar no botão "Ver tarefa" temos acesso à sua informação, e é permitido seguir para o próximo passo. É nesta etapa, representada na figura 53, que *o admissor de candidaturas a apoios* analisa os dados submetidos pelo munícipe e decide se a candidatura é admitida ou não.

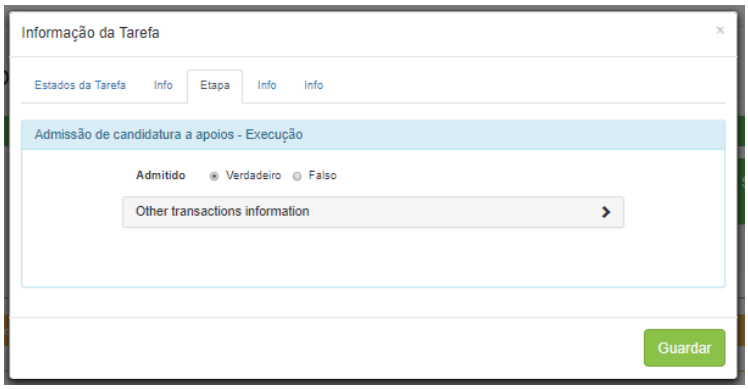
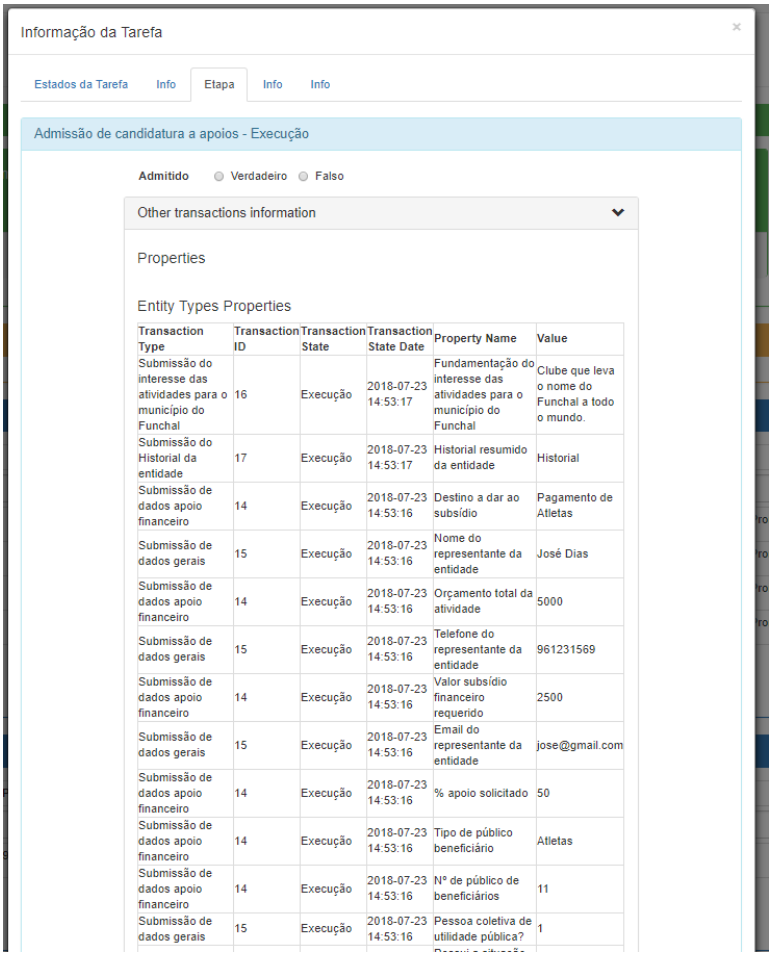


Figura 53 – Admissão de candidatura a apoios

É na aba "*Other transactions informations*" que é possível visualizar os dados submetidos pelo munícipe. Isto é possível pois quando a propriedade "Admitido" foi criada foi especificado no campo "Tipo_entidade_info" os tipos de entidade "Dados apoio financeiro", "Dados gerais", "Fundamentação do interesse das atividades para o município do Funchal" e "Historial da entidade". Assim esta propriedade consegue ter acesso a todos esses dados. Se quiséssemos que uma propriedade tivesse acesso apenas a uma propriedade específica ou mais, bastava selecionar essa(s) propriedade(s) no campo "Propriedade_info". Na figura 54 encontram-se parte desses dados, que são apresentados de acordo com a opção *accordion*.



Transaction Type	Transaction ID	Transaction State	Transaction State Date	Property Name	Value
Submissão do interesse das atividades para o município do Funchal	16	Execução	2018-07-23 14:53:17	Fundamentação do interesse das atividades para o município do Funchal	Clube que leva o nome do Funchal a todo o mundo.
Submissão do Historial da entidade	17	Execução	2018-07-23 14:53:17	Historial resumido da entidade	Historial
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	Destino a dar ao subsídio	Pagamento de Atletas
Submissão de dados gerais	15	Execução	2018-07-23 14:53:16	Nome do representante da entidade	José Dias
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	Orçamento total da atividade	5000
Submissão de dados gerais	15	Execução	2018-07-23 14:53:16	Telefone do representante da entidade	961231569
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	Valor subsídio financeiro requerido	2500
Submissão de dados gerais	15	Execução	2018-07-23 14:53:16	Email do representante da entidade	jose@gmail.com
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	% apoio solicitado	50
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	Tipo de público beneficiário	Atletas
Submissão de dados apoio financeiro	14	Execução	2018-07-23 14:53:16	Nº de público de beneficiários	11
Submissão de dados gerais	15	Execução	2018-07-23 14:53:16	Pessoa coletiva de utilidade pública?	1

Figura 54 - Visualização dos dados submetidos pelo munícipe

Supondo que o *admissor de candidaturas a apoios* aprova a candidatura, automaticamente é efetuado o pedido do tipo de transação "Seleção de candidaturas para apoios". Esse pedido fica registado no painel das tarefas do iniciador, que pode ser consultado na figura 55.

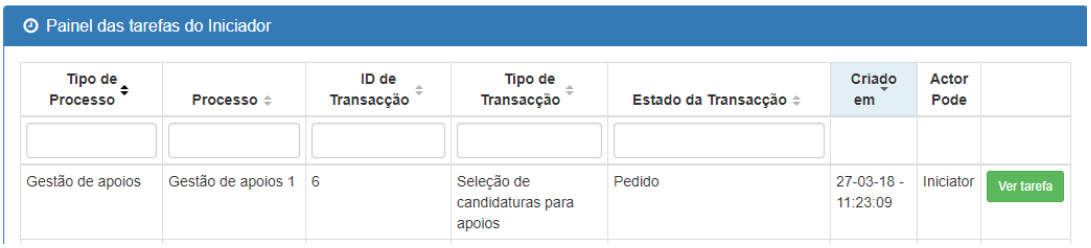


Figura 55 - Painel das tarefas do iniciador (admissor de candidaturas a apoios)

Seguidamente, cabe ao *selecionador de candidaturas* decidir se a candidatura é selecionada ou não. Para isso o *selecionador de candidaturas* acede ao seu painel de tarefas de execução, representado na figura 56, e clica em "Ver tarefa".

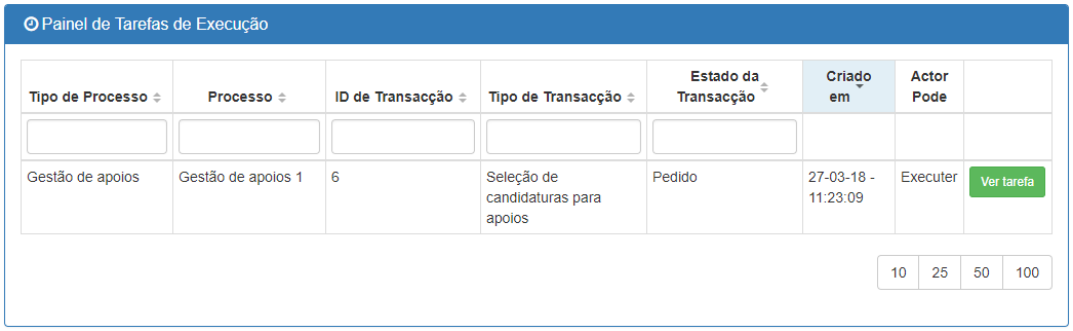


Figura 56 - Painel de tarefas de execução (selecionador de candidaturas)

Ao avançar para o próximo passo surge o *modal* que está na figura 57.

Figura 57 - Seleção de candidaturas a apoios

Tal como anteriormente, o *selecionador de candidatura* tem acesso a todos os dados inseridos pelo munícipe. É nesta fase que é verificado se as candidaturas são de interesse para a Câmara Municipal do Funchal ou não. Assumindo que a candidatura é de interesse e é selecionada, automaticamente o pedido de "Decisão sobre candidatura a apoios" é efetuado.

Como pode ser visualizado na figura 58 no painel de tarefas de execução do *decisor sobre atribuição de apoios* já aparece o tipo de transação.

Figura 58 - Painel de tarefas de execução (decisor sobre atribuição de apoios)

Finalmente o *decisor sobre atribuição de apoios* irá olhar para a candidatura e decidir se o apoio será atribuído ou não. Caso seja aprovado, é necessário especificar o valor que será atribuído. O *modal* de decisão sobre atribuição de apoios está presente na figura 59.

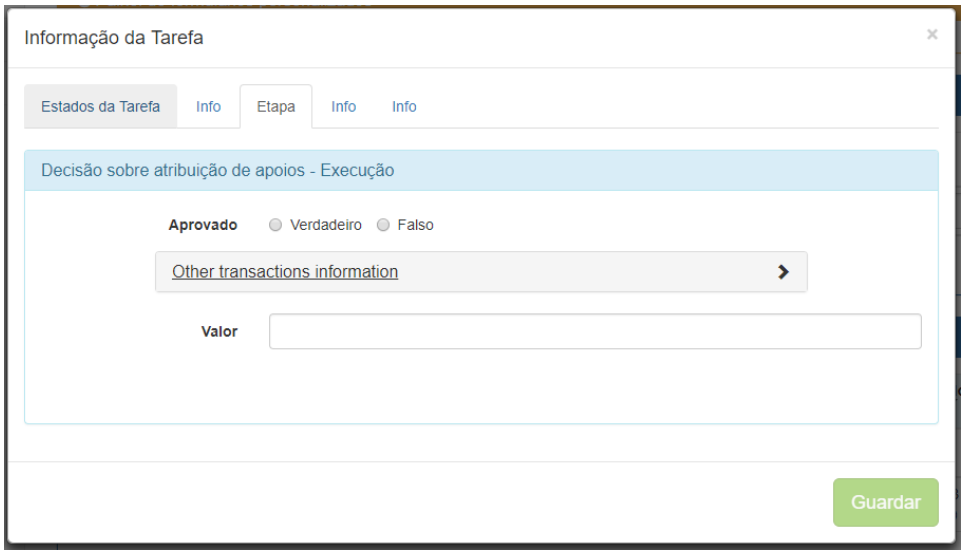


Figura 59 - Decisão sobre atribuição de apoios

Depois de todo esse processo, o munícipe é avisado se o apoio lhe foi atribuído ou não, através de uma notificação.

5.5.2 Gestão de Relações

O componente Gestão de Relações permite gerir os tipos de relações, ou seja, permite inserir, editar, visualizar e remover tipos de relações.

Na página inicial do componente Gestão de Relações é possível visualizar uma tabela com os tipos de relações existentes. Um tipo de relação é sempre constituído por dois tipos de entidades, e está associado a um tipo de transação. Na figura 60 é apresentada a página inicial deste componente.

Figura 60 - Página inicial Gestão de Relações

Como podemos observar na figura 60, temos um tipo de relação entre os tipos de entidade Transporte e Entidade, o que significa que uma entidade pode fazer vários pedidos de transporte e um transporte pode ser pedido por vários tipos de entidades.

Para adicionar um novo tipo de relação basta clicar no botão "Adicionar tipos de relação" e surge o formulário que está na figura 61. Para inserir um novo tipo de relação, é necessário preencher obrigatoriamente todos os campos, ou seja, o nome do novo tipo de relação, os tipos de entidades pelo qual o tipo de relação é constituído, o tipo de transação ao qual o tipo de relação está associado e por fim o seu estado.

Figura 61 - Formulário para inserir tipos de relações

Para editar um certo tipo de relação apenas é necessário clicar no botão “Editar” e surge o formulário que está na figura 62. Todos os campos são editáveis.

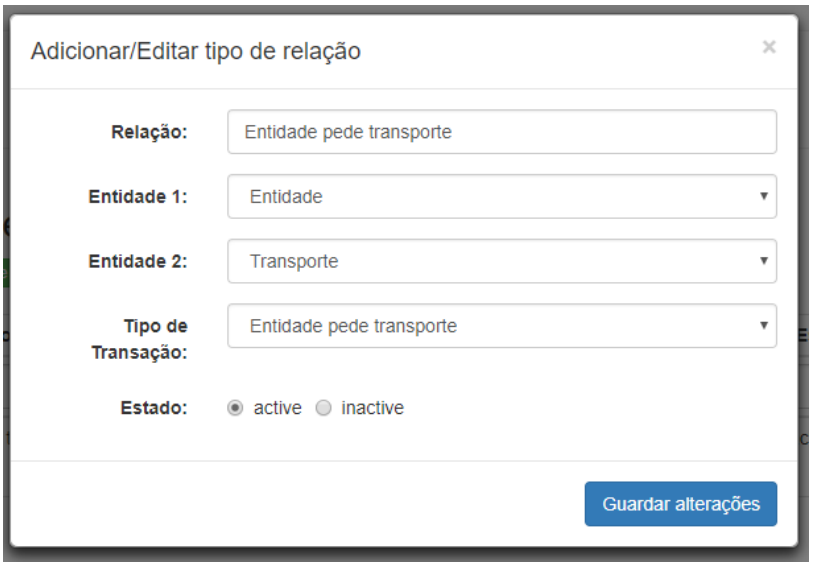


Figura 62 - Formulário para editar um certo tipo de relação

5.5.3 Gestão de Propriedades

O componente Gestão de Propriedades permite gerir as propriedades das entidades e as propriedades das relações.

5.5.3.1 Propriedades das Entidades

Na página inicial das Propriedades das Entidades é possível visualizar uma tabela com os tipos de entidades e as suas propriedades associadas. Nesta tabela, devido à utilização do módulo *ng-table* do AngularJS, é possível efetuar a ordenação, paginação e aplicar filtros de pesquisa. A ordenação pode ser realizada em todos os campos e os filtros de pesquisa podem ser efetuados no nome da propriedade e no tipo de entidade. Na figura 63 está apresentada a página inicial deste componente.

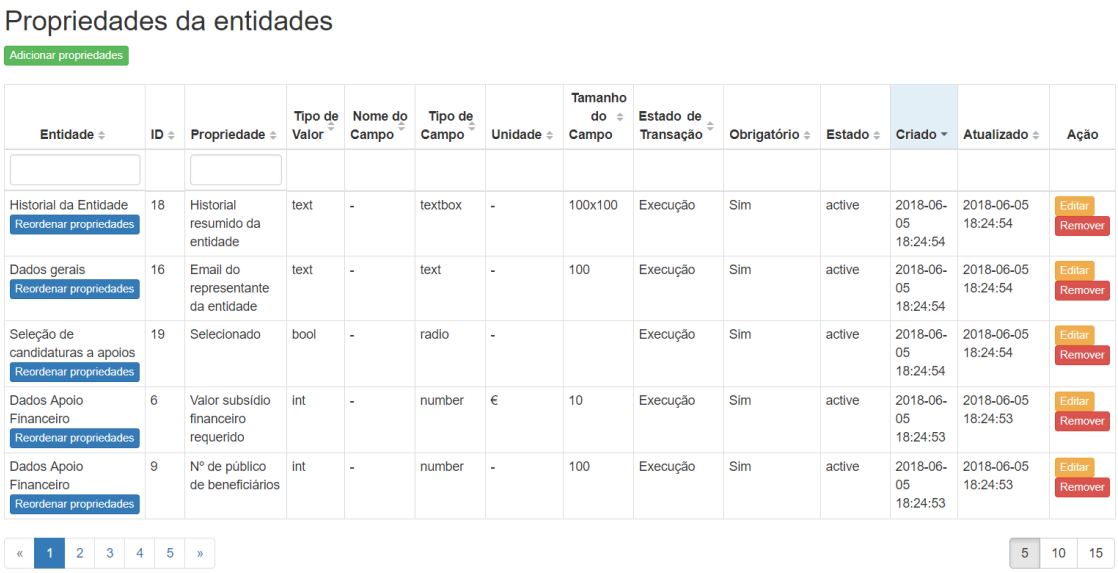


Figura 63 - Página inicial Propriedades das Entidades

Para adicionar uma nova propriedade num tipo de entidade basta clicar no botão "Adicionar propriedades" e surge o formulário que está na figura 64.

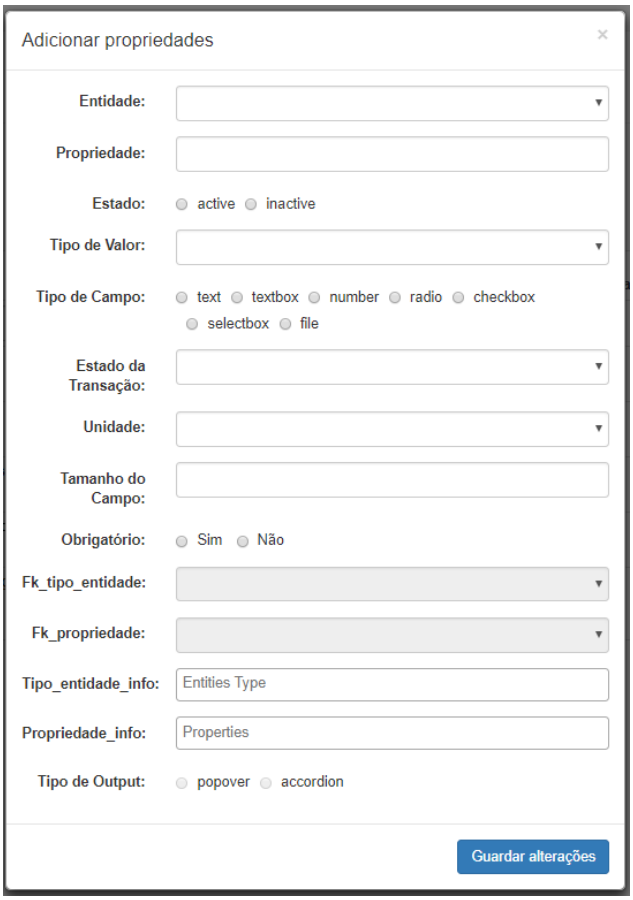


Figura 64 – Formulário para inserir propriedades num tipo de entidade

No formulário apresentado na figura 64 existem campos que são de preenchimento obrigatório e outros que não.

Os campos obrigatórios são "Entidade" em que deve ser selecionado o tipo de entidade onde a nova propriedade será introduzida, o campo "Propriedade" que é o nome da nova propriedade, o campo "Estado" que pode ser ativo ou inativo, o campo "Tipo de Valor" que se refere ao tipo de valor que o campo do formulário pode aceitar (por exemplo, se pode aceitar texto, inteiros, booleanos, etc.), o campo "Tipo de Campo" que se refere ao tipo de campo do formulário (por exemplo, se o campo é um *input* de texto, se é um *selectbox*, *radio*, *checkbox*, etc), o campo "Estado da Transação" que diz respeito ao estado da transação ao qual a propriedade se encontra no formulário e por fim o campo "Obrigatório" em que deve ser indicado se o campo do formulário é de preenchimento obrigatório ou não.

Ao preencher apenas estes campos já é possível criar as propriedades mais simples. Para esclarecer como as propriedades são criadas, consideremos o formulário da figura 65.

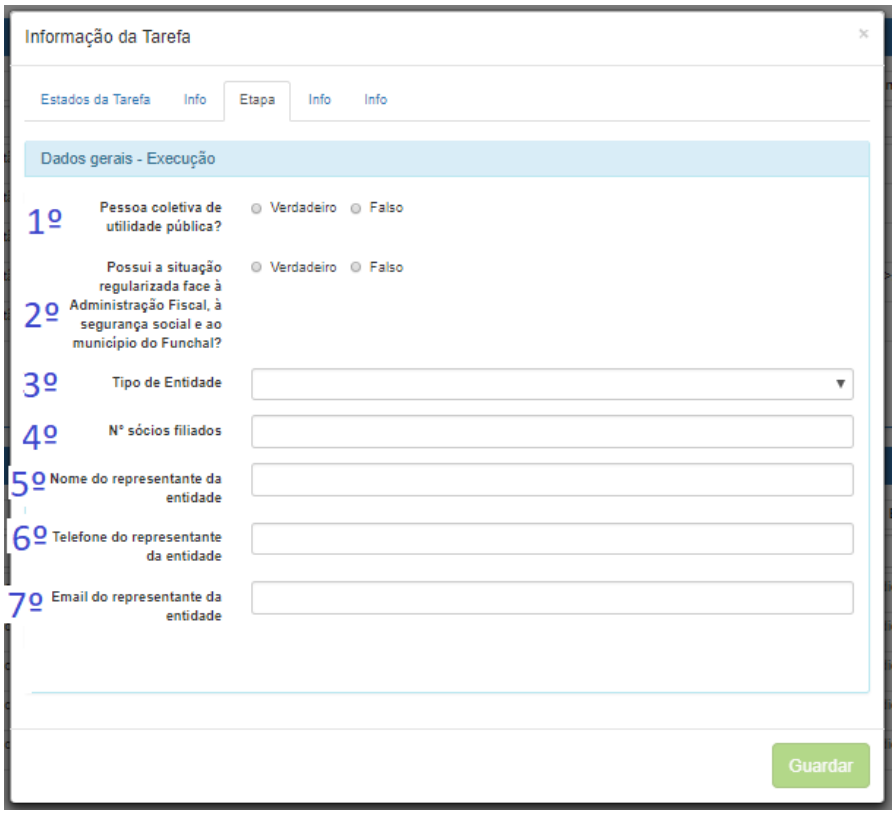


Figura 65 - Formulário "Submissão de dados apoio financeiro""

Observando o formulário, podemos constatar que este é constituído por diferentes propriedades. Para conseguir esse resultado, as propriedades tiveram de ser inseridas no tipo de entidade “Dados Gerais”, no estado de transação “Execução”, com o estado “Ativo” e eram de preenchimento obrigatório. O que as diferenciou foi os campos tipo de valor e tipo de campo.

Por exemplo, como podemos observar no formulário representado na figura 65, o primeiro e segundo campo a preencher só tem duas opções como resposta: verdadeiro ou falso. Para este caso, o tipo de valor da propriedade foi “bool” (que significa booleano - tipo de dado primitivo que só possui dois valores: 0 ou 1, isto é, verdadeiro ou falso), e o tipo de campo foi “radio” uma vez que o utilizador só pode selecionar apenas uma opção das duas opções apresentadas.

No terceiro campo do formulário da figura 65, queríamos que apenas fosse possível escolher uma das opções apresentadas. Portanto, o tipo de valor foi “enum” e o tipo de campo foi “*selectbox*”. Uma vez que o tipo de valor foi “enum”, para aparecer as opções pretendidas no *selectbox,* tivemos que inserir valores permitidos para essa propriedade.

Já no quarto e sexto campo do formulário da figura 65, queríamos que apenas fosse possível digitar números inteiros. Dessa forma, ao introduzir as propriedades, em tipo de valor colocamos “*int*” e no tipo de campo colocamos “*number*”. Assim, surge um *input* do tipo *number* (campo para inserir números) em que só é possível digitar números inteiros.

Por fim, no quinto e sétimo campo do formulário da figura 65, queríamos que apenas fosse possível introduzir caracteres. Para isso, quando as propriedades foram introduzidas, no tipo de valor escolhemos "*text*" e no tipo de campo também escolhemos "*text*". Assim, surgiu um *input* do tipo texto para inserir caracteres.

Continuando com a explicação dos restantes campos do formulário 64, os campos opcionais são "Unidade", "Tipo_entidade_info" e "Propriedade_info". O campo "Unidade" deve ser preenchido apenas se a propriedade possuir unidades, como por exemplo cm, m, €. Os campos "Tipo_entidade_info" e "Propriedade_info" são utilizados caso se queira ter acesso a dados de tipos de entidades ou propriedades, respetivamente.

No campo "Tipo_entidade_info" é possível selecionar múltiplos tipos de entidades e, caso não se queira ter acesso aos dados de um tipo de entidade completo, no campo "Propriedade_info" é possível selecionar propriedades específicas de tipos de entidades diferentes.

Para implementar a multiselecção foi utilizado a biblioteca *SELECT2* que permite um selectbox personalizável, com suporte para pesquisa e múltipla seleção.

Na figura 66 está exposto um exemplo do campo "Tipo_entidade_info" com vários tipos de entidades selecionadas.

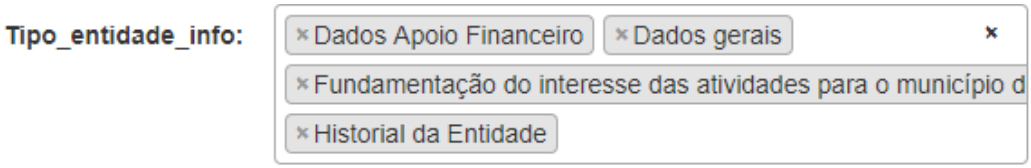


Figura 66 - Campo "Tipo_entidade_info"

Já na figura 67 é exibido um exemplo do campo "Propriedade_info" com duas propriedades selecionadas. Neste caso as propriedades são agrupadas por tipo de entidade (que estão a negrito e que não podem ser selecionadas) para distinguir a que tipo de entidade as propriedades pertencem.

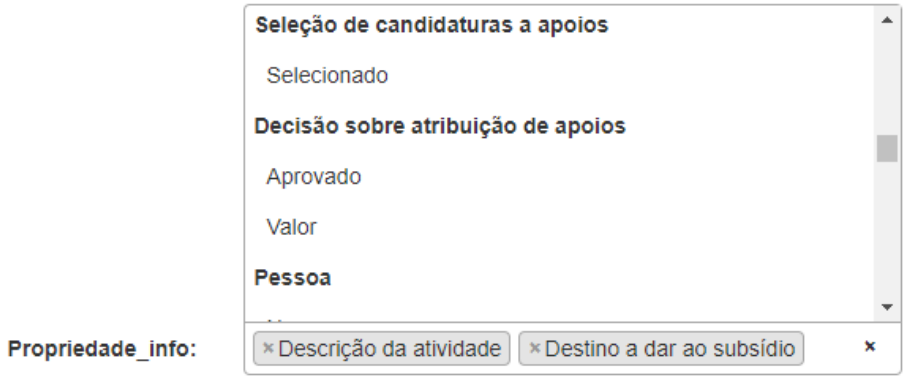


Figura 67 - Campo "Propriedade_info"

Estes campos são fundamentais uma vez que quando os formulários são gerados pelo *dashboard*, por vezes é útil apresentar informação relacionada. Por exemplo, como

já tinha sido apresentado na explicação do *dashboard*, no caso de "Apoios para o desenvolvimento de atividades de interesse municipal", em determinado momento o *admissor de candidaturas a apoios* necessita de ter acesso aos dados introduzidos pelo munícipe para analisar os dados e decidir se a candidatura é admitida ou não. É devido ao campo "Tipo_entidade_info" que os dados do munícipe ficam disponíveis para consulta pelo *admissor de candidaturas a apoios,* pois quando a propriedade "Admitido" foi criada foi especificado no campo "Tipo_entidade_info" os tipos de entidade "Dados apoio financeiro", "Dados gerais", "Fundamentação do interesse das atividades para o município do Funchal" e "Historial da entidade". Como exemplo da relevância do campo "Tipo_entidade_info", é apresentado novamente a imagem que se encontra na figura 68, em que na Aba "*Other transactions informations*" estão os dados que se pretende visualizar.

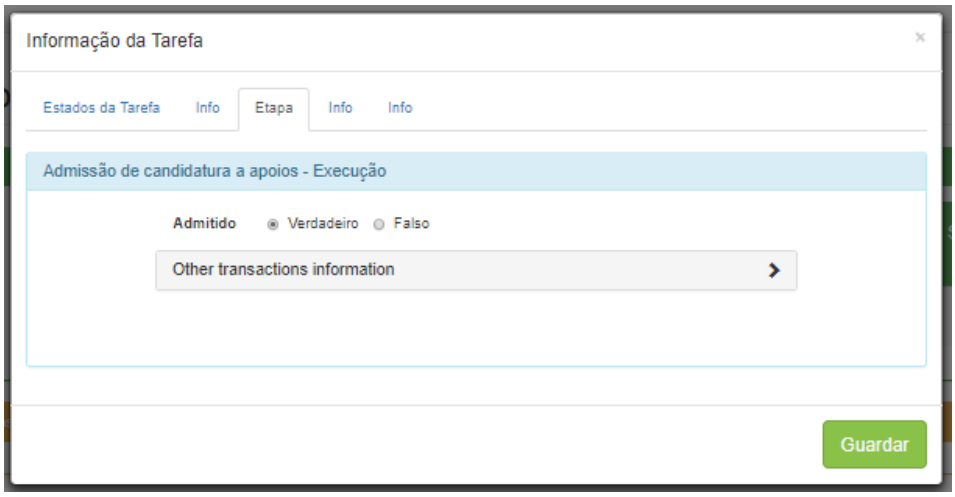


Figura 68 - Dados disponível na aba "Other transaction informations"

Prosseguindo com a explicação dos campos do formulário da figura 64, também existem campos em que a sua obrigatoriedade depende de outros. Por exemplo:

- Caso o "Tipo de Campo" seja *text* ou *textbox*, o "Tamanho do Campo" deve ser preenchido;
- Caso o "Tipo de valor" seja *prop_ref*, o "Fk_propriedade" é obrigatório;
- Caso os campos "Tipo_entidade_info" ou "Propriedade_info" sejam preenchidos, o campo "Tipo de Output" é obrigatório.

O campo "Tamanho do Campo" refere-se ao tamanho do campo no formulário, já o campo "Fk_propriedade*"* serve para indicar que uma propriedade referencia outra propriedade. O campo "Fk_tipo_entidade*"* apenas serve como filtro caso o tipo de valor seja *prop_ref*. Foi decidido filtrar pelo tipo de entidade porque propriedades de tipos de entidades diferentes podem ter o mesmo nome, e se não filtrássemos pelo tipo de entidade, não iria ser possível saber a que tipo de entidade a propriedade pertencia.

Em último lugar, o campo "Tipo de Output" refere-se ao modo de apresentação dos dados a que a propriedade tem acesso. Decidiu-se optar pela opção *accordion* que

fornece uma lista de itens que são omitidos ou expandidos ao clicar no cabeçalho (ver figuras 53 e 54).

Em relação aos campos de preenchimento obrigatório foram efetuadas validações, que podem ser observadas na figura 69.

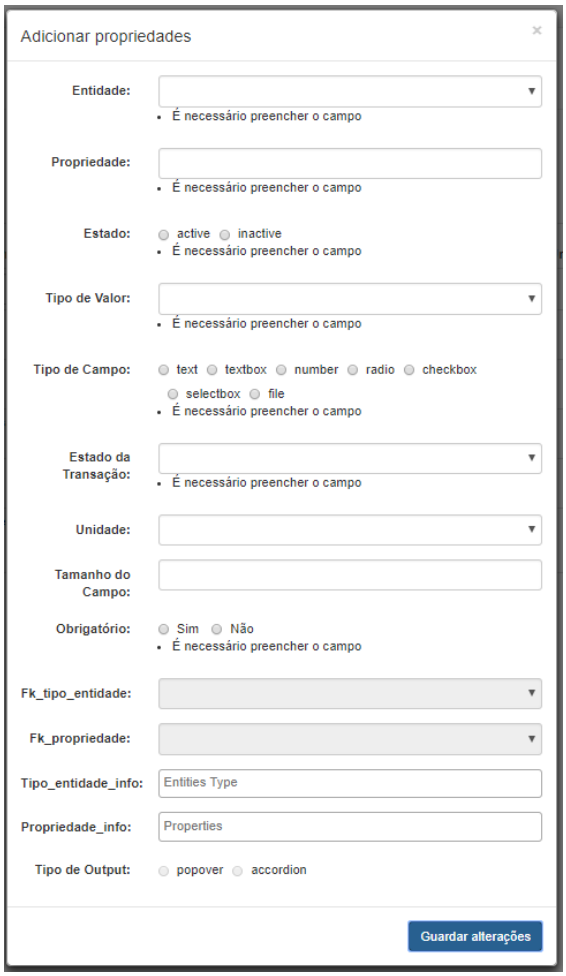


Figura 69 – Exemplos de validações nos campos do formulário para inserir propriedades

Também foram efetuadas outras validações devido aos campos "Tipo de Valor" e "Tipo de Campo". Para cada tipo de valor, o tipo de campo só pode tomar determinados valores. Estes podem ser consultados na tabela 5.

Tipo de Valor	*Tipo de Campo*
text	*text, textbox*
bool	*radio, selectbox*
int e double	*number*
enum	*radio, checkbox, selectbox*
prop_ref	*selectbox*
file	*file*

Tabela 5 – Tipos de campo permitidos conforme tipo de valor

À medida que vamos adicionando propriedades podemos alterar a ordem em que os campos do formulário serão exibidos. Para isso basta clicar no botão "Reordenar

propriedades". Por exemplo, se for pretendido alterar a ordem das propriedades do tipo de entidade "Dados apoio Financeiro" surge o *modal* da figura 70.

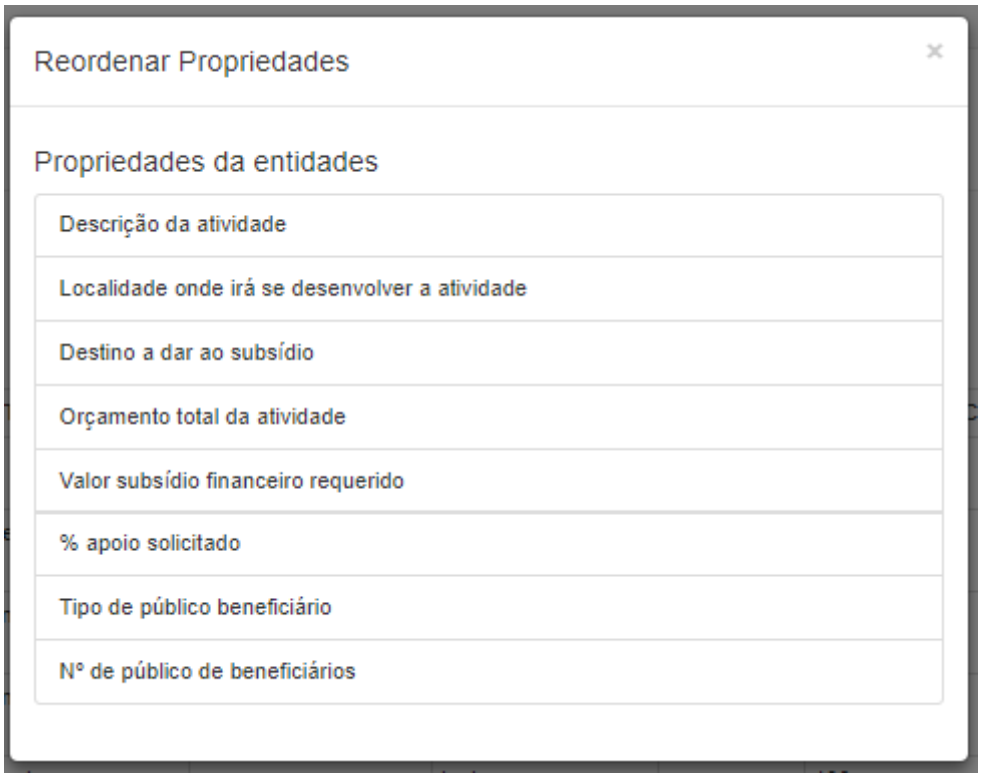


Figura 70 - Reordenar propriedades

Para reordenar as propriedades foi utilizada a funcionalidade *drag and drop*. Portanto, para reordená-las basta clicar e manter o botão do rato pressionado, para "agarrar" a propriedade, arrastá-la (*drag*) para a posição desejada e por fim soltar o botão pressionado (*drop*). Imediatamente após a reordenação as alterações são guardadas na base de dados. Para implementar o *drag and drop*, foram incluídas todas as bibliotecas necessárias via *scripts* CDN (*Content Delivery Network*) e foi utilizado o módulo jQuery *ui-sortable* para AngularJS.

Se for pretendido alterar os dados de uma determinada propriedade basta clicar no respetivo botão "Editar" que surge o formulário de edição. Este formulário de edição é semelhante ao de inserção, com a única diferença que o de edição já possui os campos pré-preenchidos.

5.5.3.2 Propriedades das Relações

Na página inicial das Propriedades das Relações é possível visualizar uma tabela com os tipos de relações e as suas propriedades associadas. Tal como na página de Propriedades das Entidades é possível efetuar a ordenação que pode ser realizada em todos os campos, a paginação e aplicar filtros de pesquisa no nome do tipo de relação e no nome da propriedade. Na figura 71 está apresentada a página inicial deste

componente.

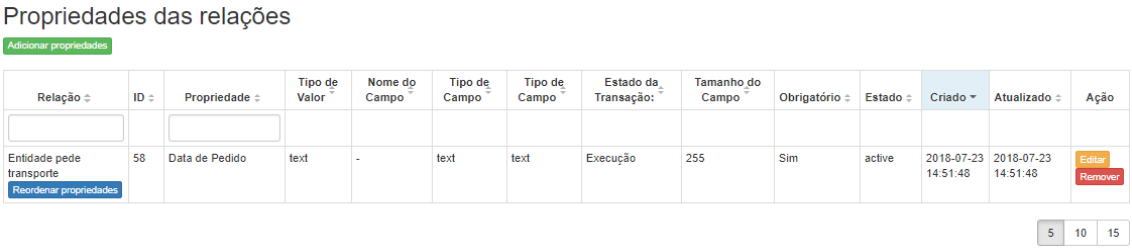


Figura 71 – Página inicial Propriedades das Relações

Para adicionar uma nova propriedade num tipo de relação basta clicar no botão "Adicionar propriedades" e surge o formulário que está na figura 72. Neste contexto, para adicionar uma nova propriedade com sucesso é necessário selecionar o tipo de relação em que a propriedade irá pertencer, o nome da propriedade, o seu estado, o tipo de valor, o tipo de campo, o estado da transação ao qual a propriedade irá estar associada, a unidade (opcional), o tamanho do campo e a sua obrigatoriedade. Tal como no caso das propriedades das entidades são efetuadas validações nos campos e também é permitido reordenar as propriedades.

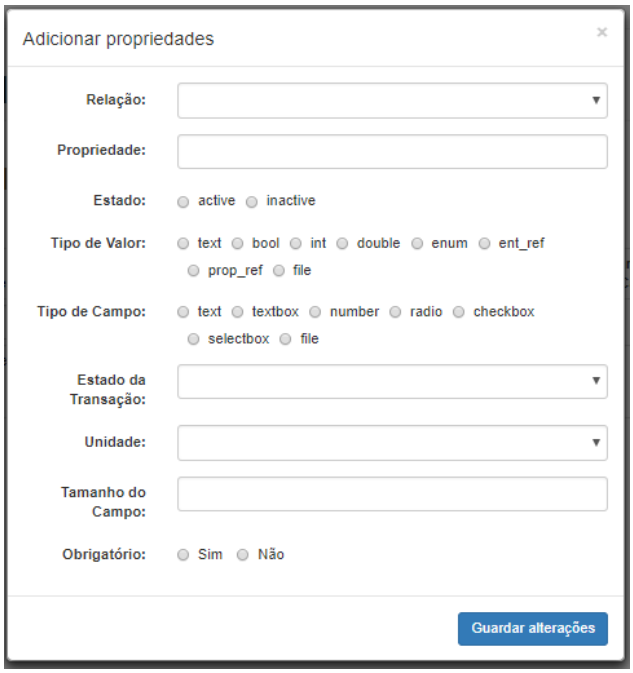


Figura 72 - Formulário para inserir propriedades num tipo de relação

Se for pretendido alterar os dados de uma determinada propriedade basta clicar no respetivo botão "Editar" que surge o formulário de edição. Este formulário de edição é semelhante ao de inserção, com a única diferença que o de edição já possui os campos pré-preenchidos.

5.5.4 Gestão de Pesquisas

No componente Gestão de Pesquisas para além de ser possível construir pesquisas dinâmicas, também é permitido guardá-las e exportar os seus resultados para formato Excel.

5.5.4.1 Pesquisa Dinâmica

O objetivo da Pesquisa Dinâmica é a construção de pesquisas, em que o administrador seleciona as propriedades, valores e operadores num tipo de entidade através de uma interface gráfica. Na primeira página da Pesquisa Dinâmica, que está representada na figura 73, são listados todos os tipos de entidade existentes na base de dados.

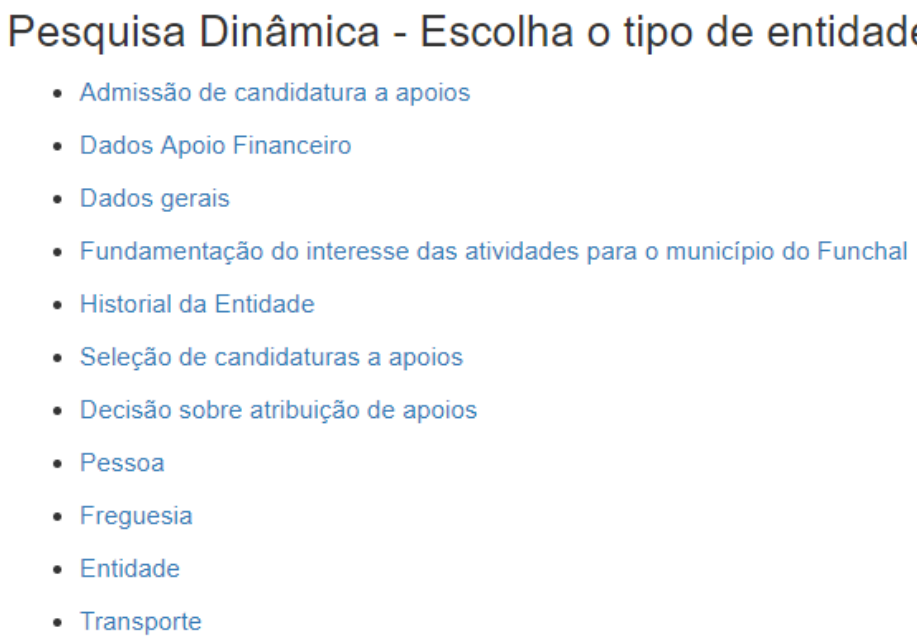


Figura 73 - Página Inicial Pesquisa Dinâmica

O primeiro passo para efetuar a pesquisa dinâmica é escolher um tipo entidade. Depois de escolhido o tipo de entidade pretendido somos direcionados para a página onde podemos especificar os campos a pesquisar. Para efetuar a pesquisa surgem tabelas que permitem criar quatro tipos de "filtros" diferentes.

A primeira tabela contém todas as propriedades do tipo de entidade escolhido, a segunda tabela inclui as propriedades dos tipos de entidades que contenham pelo menos uma propriedade que referencie uma propriedade do tipo de entidade selecionado, a terceira tabela é composta pelas propriedades dos tipos de relação em que o tipo de entidade escolhido está presente, e por fim a quarta tabela exibe os tipos de entidades (e respetivas propriedades) que se relacionam com o tipo de entidade selecionado.

Em termos de pesquisa, esta pode ser efetuada selecionando propriedades apenas da primeira tabela, ou então, podemos efetuar pesquisas cruzando informação de duas tabelas (por exemplo, a primeira tabela com a segunda, a primeira com a terceira, e a

primeira com a quarta). Face a isso foram implementadas validações, em que para selecionar propriedades da segunda, terceira ou quarta tabela é obrigatório selecionar pelo menos uma propriedade da primeira.

Em cada uma destas tabelas é apresentada uma *checkbox* à frente de cada propriedade bem como um campo onde o utilizador poderá restringir os valores para essa mesma propriedade. Este campo adapta-se de acordo com o tipo de valor da propriedade. Por exemplo, se o tipo for *enum* aparece uma *selectbox* com a lista dos valores permitidos. Se for *int* ou *double*, o campo deve ser procedido por uma *selectbox* que contenha os operadores menor, maior, igual, diferente e aproximado (<, >, =, !=, ~ respetivamente) e apenas é permitido inserir números. É de referir que este componente é totalmente dinâmico indo buscar à base de dados na tabela *operator* a lista de operadores disponíveis.

As tabelas são desenvolvidas deste modo para que seja possível restringir a lista de instâncias do tipo de entidade selecionado com um conjunto de parâmetros aplicados às propriedades escolhidas, sendo que deve ser dada ao utilizador a capacidade de escolher trios de: propriedade, operador e valor.

Para uma melhor compreensão será descrito como a pesquisa é efetuada de acordo com as propriedades selecionadas nas várias tabelas, e será apresentado um exemplo para cada caso.

5.5.4.1.1 Pesquisa utilizando apenas a 1º tabela

Como já tinha sido mencionado podemos realizar uma pesquisa selecionando apenas propriedades da primeira tabela.

Caso o tipo de entidade “Dados Gerais” seja selecionado, surge a página que está representada na figura 74. Como não existem propriedades de entidades que referenciem uma propriedade do tipo de entidade selecionado, e como o tipo de entidade selecionado não está envolvido em nenhum tipo de relação apenas surge a tabela com a lista de propriedades do tipo de entidade “Dados Gerais”. Nesta tabela são apresentadas propriedades do tipo *radio*, *enum*, *int* e *text*.

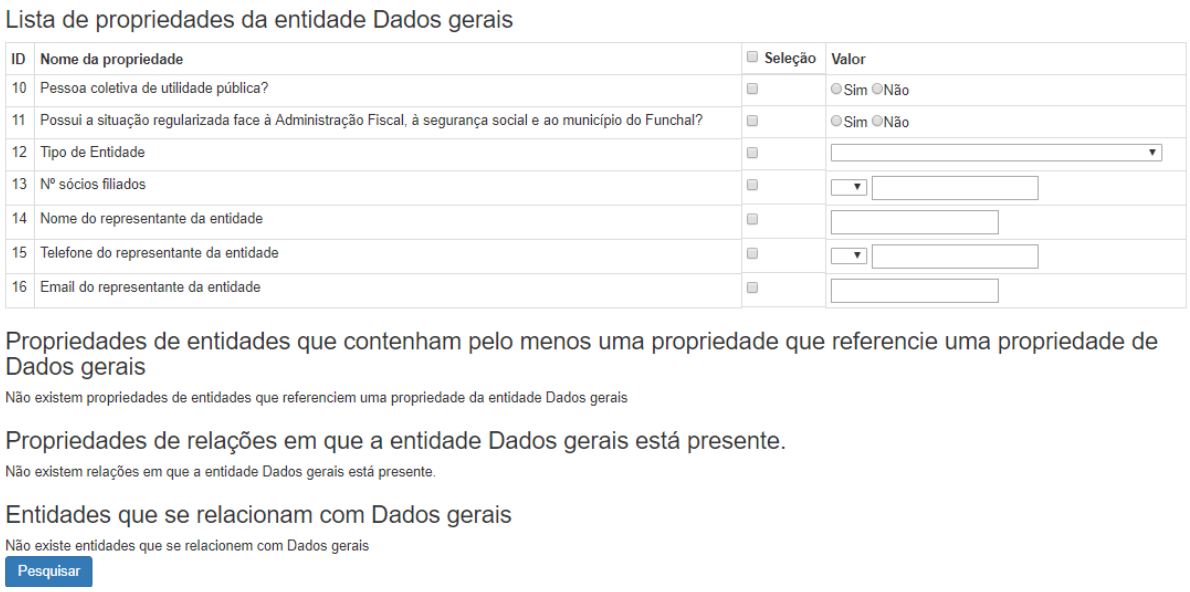


Figura 74 – 1º tabela da Pesquisa Dinâmica

Caso se pretenda efetuar uma pesquisa com todas as propriedades, é possível selecioná-las todas de uma vez ao clicar na *checkbox* que se encontra no cabeçalho da tabela.

Em termos de pesquisa, se selecionarmos, por exemplo, a propriedade "Nome do representante da entidade" e não especificarmos nenhum valor, não é aplicado nenhum filtro e deste modo é permitido visualizar representantes da entidade com qualquer nome. Caso se selecione a propriedade "Nº de Sócios" e se especificarmos o operador igual a > e o valor igual a 10 vamos visualizar apenas as instâncias cujo nº de sócios seja maior que 10.

Após determinar o conjunto de trios, é construída uma *query* SQL dinamicamente que executa a pesquisa construída e faz o output para o utilizador. Como resultado da pesquisa surge uma tabela em que o cabeçalho é composto pelas propriedades selecionadas na pesquisa, e no corpo da tabela surge os valores que correspondem à pesquisa efetuada. Quanto mais propriedades selecionarmos e mais valores inserimos no formulário mais filtrada será a pesquisa e mais específicos serão os resultados. O resultado da pesquisa do exemplo relativamente ao número de sócios é apresentado na figura 75.

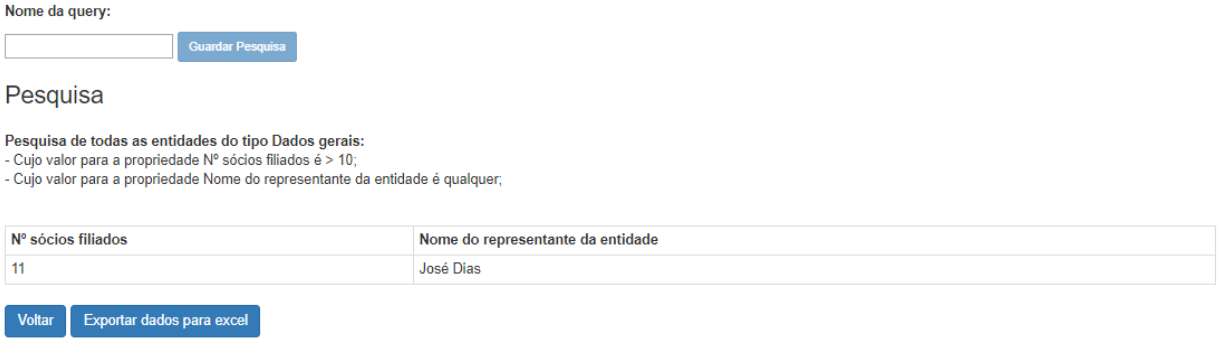


Figura 75 – Exemplo de resultado da pesquisa na 1º tabela

O procedimento para efetuar a pesquisa nesta tabela e chegar a este resultado é de acordo com as propriedades que são selecionadas e com os valores inseridos (ou não) pelo utilizador. Através disso obtém-se os identificadores das instâncias das entidades, e para cada uma dessas instâncias, são retornadas as suas propriedades e valores para apresentar na tabela.

5.5.4.1.2 Pesquisa cruzando informações da 1º e 2º tabela

Caso se queira cruzar informações da primeira tabela com a segunda, na página inicial da Pesquisa Dinâmica (representada na figura 73) temos de escolher o tipo de entidade "Freguesia". Logo de seguida surge a página que está representada na figura 76.

Neste caso a propriedade "Freguesia" do tipo de entidade "Pessoa" referencia a propriedade "Nome" do tipo de entidade "Freguesia", e é por esse motivo que a segunda tabela é apresentada. Neste enquadramento, a terceira e quarta tabela não são exibidas uma vez que o tipo de entidade selecionado não está envolvido em nenhum tipo de relação.

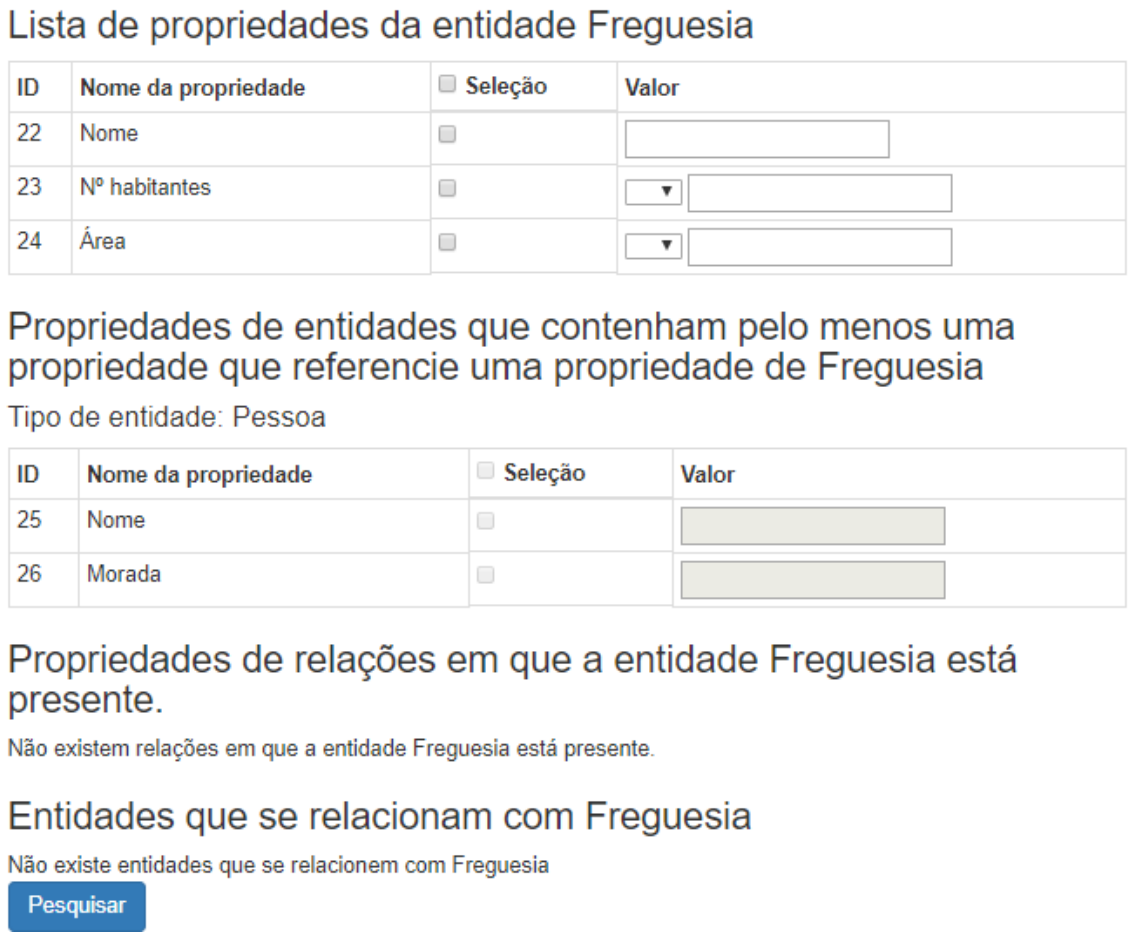


Figura 76 - 1º e 2º tabela da Pesquisa Dinâmica

Neste contexto, pode ser efetuada uma pesquisa para visualizar a que freguesia as pessoas pertencem. Assim sendo, basta selecionar a propriedade "Nome" da primeira tabela e selecionar a propriedade "Nome" da segunda. Ao realizar a pesquisa, se não for

especificado nenhum valor, iremos poder observar os resultados que estão representados na figura 77.

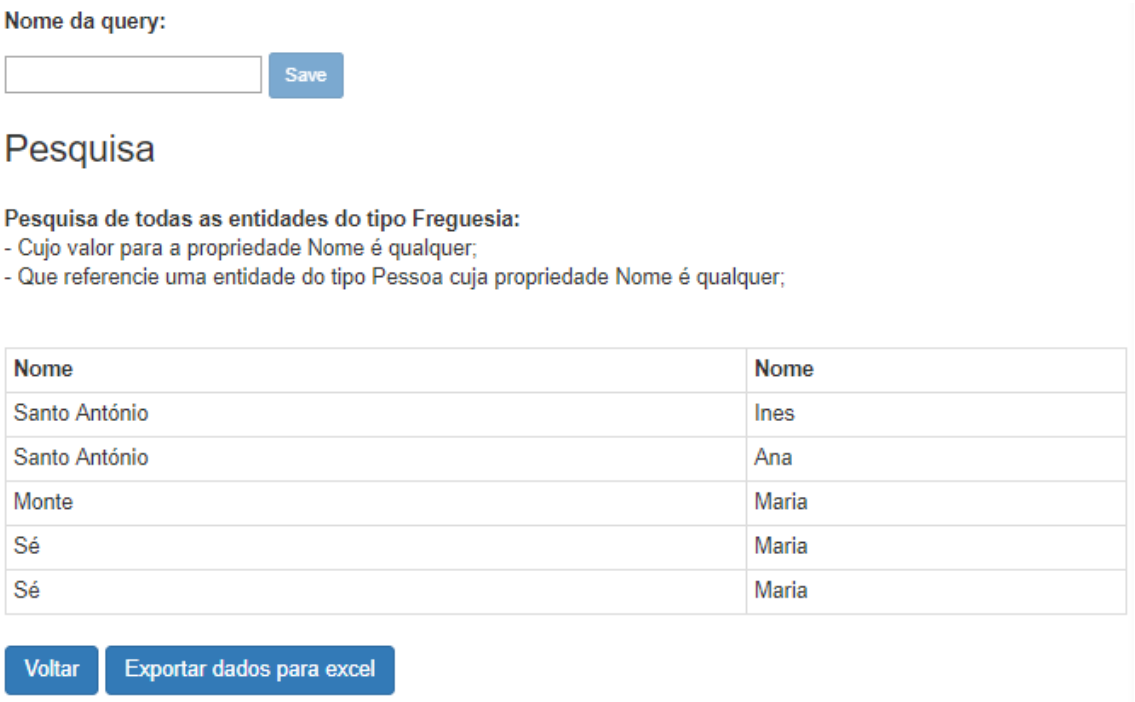


Figura 77 – Exemplo de resultado da pesquisa entre a 1º e 2º tabela

O procedimento para efetuar a pesquisa nesta tabela e chegar a este resultado passa por obter (de acordo com as propriedades, operadores e valores inseridos em ambas as tabelas) todos os identificadores das instâncias de entidades.

Depois de obter os identificadores das instâncias de entidades da primeira e da segunda tabela, esses identificadores são comparados com o objetivo de verificar se há alguma propriedade das instâncias de entidades da segunda tabela que referencie uma propriedade das instâncias de entidades da primeira tabela. De acordo com essa verificação, obtemos apenas os identificadores das instâncias de entidades que coincidem das duas tabelas e é retornado as propriedades e valores que serão apresentados no resultado final.

5.5.4.1.3 Pesquisa cruzando informações da 1º e 3º tabela

Caso se queira cruzar informações da primeira tabela com a terceira, na página inicial da Pesquisa Dinâmica (representada na figura 73), podemos selecionar qualquer tipo de entidade que esteja envolvido em algum tipo de relação. Neste exemplo podemos escolher o tipo de entidade “Transporte” ou “Entidade”. Se optarmos por “Transporte” surge a página que está representada na figura 78.

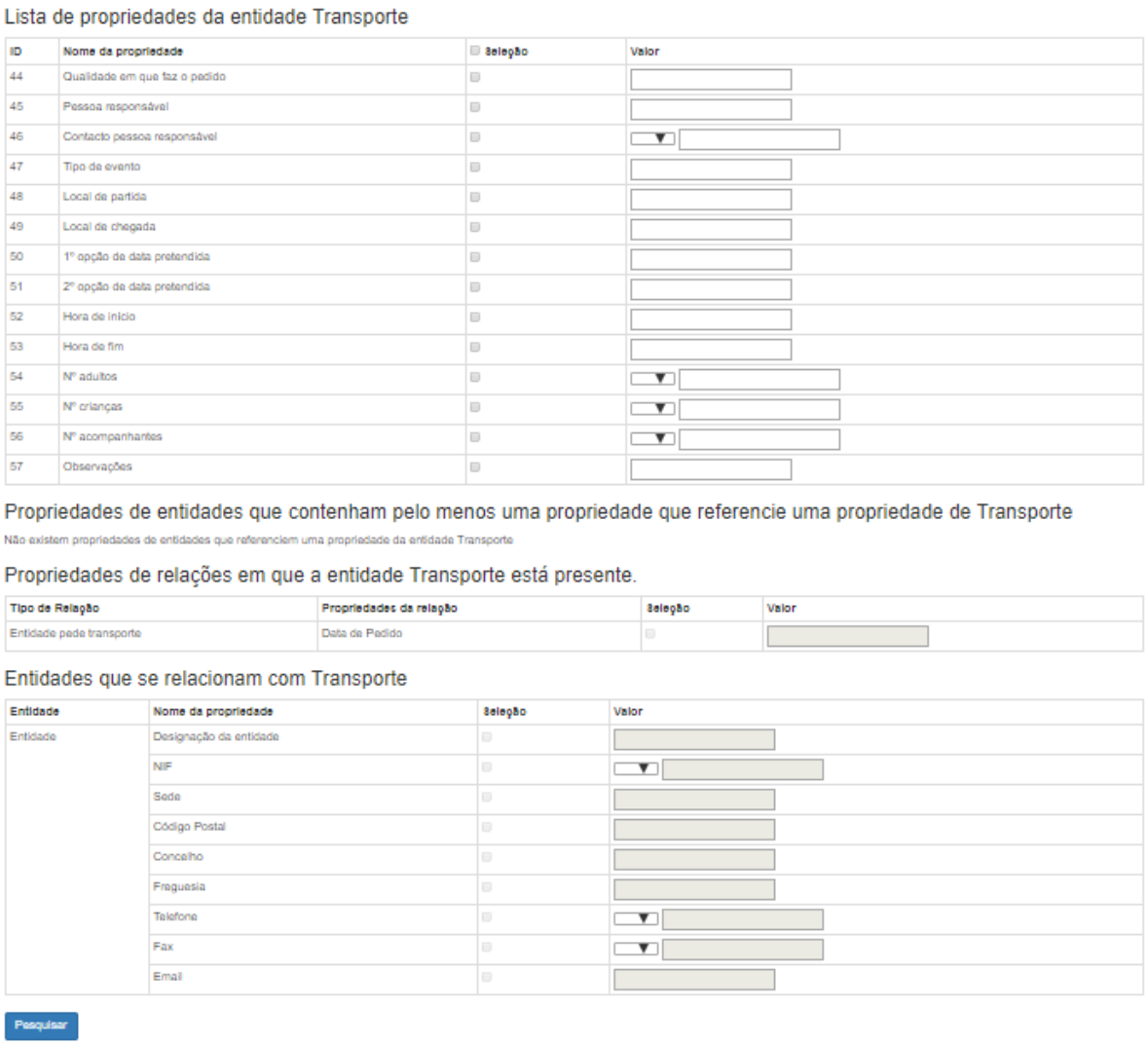


Figura 78 – 1º, 3º e 4º tabela da Pesquisa dinâmica

Neste contexto, pode ser realizada uma pesquisa para visualizar em que data os pedidos de transporte foram efetuados, ou então visualizar os pedidos de transporte que foram efetuados em alguma data específica.

Se decidirmos verificar os pedidos de transporte que foram efetuados no dia 10-02-2018, podemos selecionar todas as propriedades da primeira tabela, e da terceira tabela selecionamos a propriedade “Data de pedido” e introduzimos o valor 10-02-2018.

Ao realizar a pesquisa, podemos observar que no dia 10-02-2018 foi efetuado apenas um pedido de transporte com as características que estão presentes na figura 79.

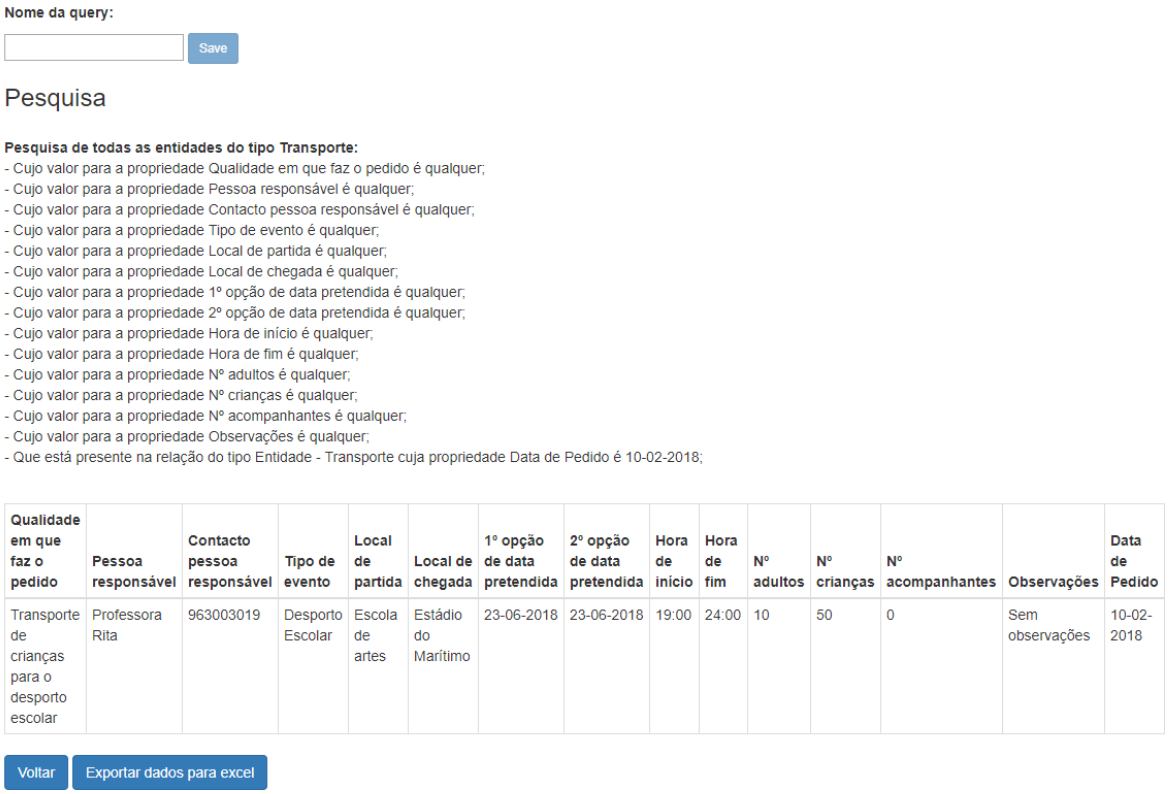


Figura 79 – Exemplo de resultado da pesquisa entre 1º e 3º tabela

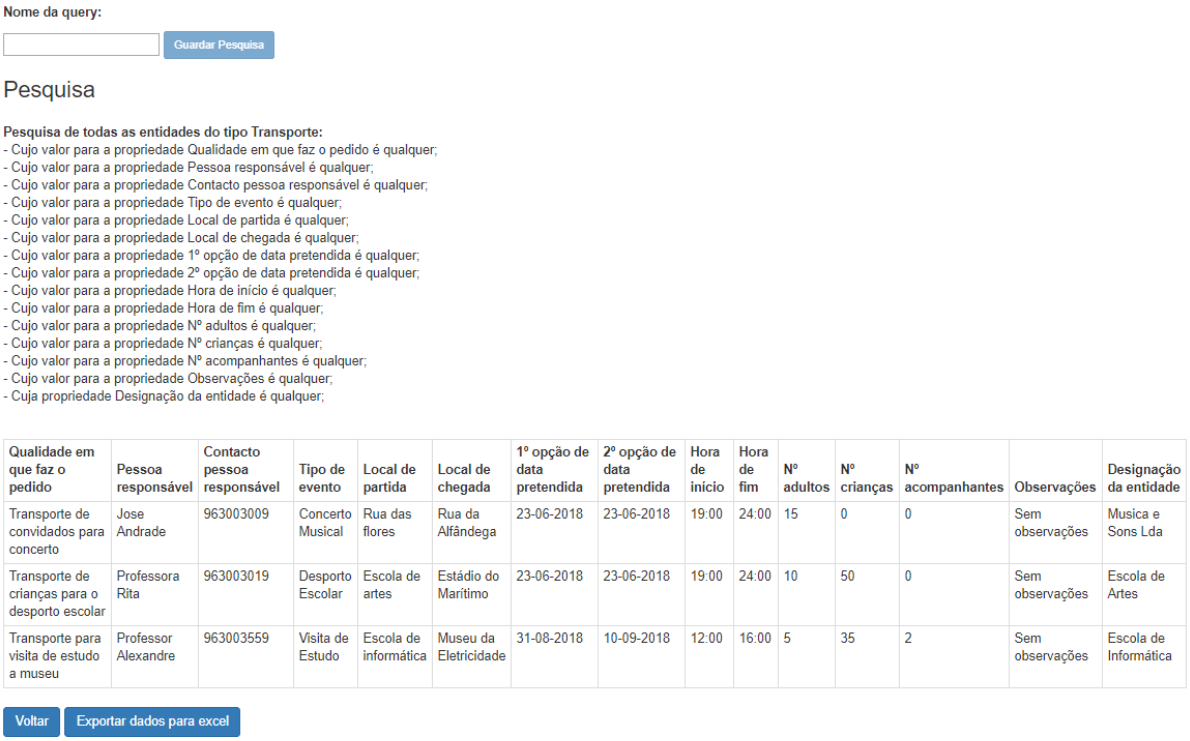
O procedimento para chegar a este resultado passa primeiro por obter os identificadores das instâncias das entidades de acordo com as propriedades, operadores e valores inseridos na primeira tabela. Seguidamente, obtém-se os identificadores das instâncias de relações de acordo com as propriedades, operadores e valores inseridos na terceira tabela, para posteriormente, para cada instância de relação, obter os identificadores das instâncias de entidades que pertencem a essa instância de relação.

Depois verificamos quais são os identificadores das instâncias de entidades em comum e de acordo com os identificadores que coincidem, retornamos as propriedades e valores para apresentar no resultado final.

5.5.4.1.4 Pesquisa cruzando informações da 1º e 4º tabela

Caso se queira cruzar informações da primeira tabela com a quarta, na página inicial da Pesquisa Dinâmica (representada na figura 73) podemos escolher novamente o tipo de entidade “Transporte”. Após selecionarmos o tipo de entidade “Transporte” surge a página que está representada na figura 78. Nesta situação pode ser realizada uma pesquisa para visualizar todas as informações dos transportes que foram efetuados pelas entidades. Para isso, todas as propriedades do tipo de entidade “Transporte” podem ser selecionadas, e da última tabela podemos selecionar a propriedade “Designação da entidade”.

Como resultado da pesquisa obtemos os dados que estão representados na figura 80.


Nome da query:

Guardar Pesquisa

Pesquisa

Pesquisa de todas as entidades do tipo Transporte:
- Cujo valor para a propriedade Qualidade em que faz o pedido é qualquer;
- Cujo valor para a propriedade Pessoa responsável é qualquer;
- Cujo valor para a propriedade Contacto pessoa responsável é qualquer;
- Cujo valor para a propriedade Tipo de evento é qualquer;
- Cujo valor para a propriedade Local de partida é qualquer;
- Cujo valor para a propriedade Local de chegada é qualquer;
- Cujo valor para a propriedade 1º opção de data pretendida é qualquer;
- Cujo valor para a propriedade 2º opção de data pretendida é qualquer;
- Cujo valor para a propriedade Hora de início é qualquer;
- Cujo valor para a propriedade Hora de fim é qualquer;
- Cujo valor para a propriedade Nº adultos é qualquer;
- Cujo valor para a propriedade Nº crianças é qualquer;
- Cujo valor para a propriedade Nº acompanhantes é qualquer;
- Cujo valor para a propriedade Observações é qualquer;
- Cuja propriedade Designação da entidade é qualquer;

Qualidade em que faz o pedido	Pessoa responsável	Contacto pessoa responsável	Tipo de evento	Local de partida	Local de chegada	1º opção de data pretendida	2º opção de data pretendida	Hora de início	Hora de fim	Nº adultos	Nº crianças	Nº acompanhantes	Observações	Designação da entidade
Transporte de convidados para concerto	Jose Andrade	963003009	Concerto Musical	Rua das flores	Rua da Alfândega	23-06-2018	23-06-2018	19:00	24:00	15	0	0	Sem observações	Musica e Sons Lda
Transporte de crianças para o desporto escolar	Professora Rita	963003019	Desporto Escolar	Escola de artes	Estádio do Marítimo	23-06-2018	23-06-2018	19:00	24:00	10	50	0	Sem observações	Escola de Artes
Transporte para visita de estudo a museu	Professor Alexandre	963003559	Visita de Estudo	Escola de informática	Museu da Eletricidade	31-08-2018	10-09-2018	12:00	16:00	5	35	2	Sem observações	Escola de Informática

Voltar | Exportar dados para excel

Figura 80 – Exemplo de resultado da pesquisa entre 1º e 4º tabela

O procedimento para chegar a este resultado passa por obter (de acordo com as propriedades, operadores e valores inseridos em ambas as tabelas) todos os identificadores das instâncias de entidades.

Posteriormente, para cada identificador de instância de entidade obtido na quarta tabela, é verificado se existe alguma relação com os identificadores das instâncias de entidades da primeira tabela. Caso exista obtemos apenas os identificadores das instâncias de entidades da primeira e da quarta tabela que se relacionam e retornamos as suas propriedades e valores para apresentar no resultado final.

5.5.4.2 Exportação de resultados para Excel

Caso se pretenda exportar os resultados da pesquisa para um formato legível pelo Microsoft Excel, na página de resultados da pesquisa existe um botão "Exportar dados para excel". Para implementar a exportação foi utilizada a biblioteca PhpSpreadsheet, que é uma biblioteca escrita em PHP puro e que fornece um conjunto de classes que permite ler e escrever em diferentes formatos, tal como o Excel e o LibreOffice Calc. Esta biblioteca foi instalada utilizando o *composer* através do comando *composer require phpoffice/phpspreadsheet* [41].

5.5.4.3 Pesquisas Guardadas

Como pode ser observado na página de resultados da pesquisa da figura 80, existe uma caixa de texto para introduzir o nome da *query*. Isto serve para guardar a pesquisa, caso se queira efetuá-la novamente noutro momento sem ter de passar pelo processo de preenchimento do formulário mais uma vez. Para guardar a pesquisa é necessário dar-lhe um nome e em seguida basta carregar no botão "Guardar Pesquisa" que a pesquisa é guardada.

Essas pesquisas/*querys* são guardadas na página Pesquisas Gravadas, que está representada na figura 81.

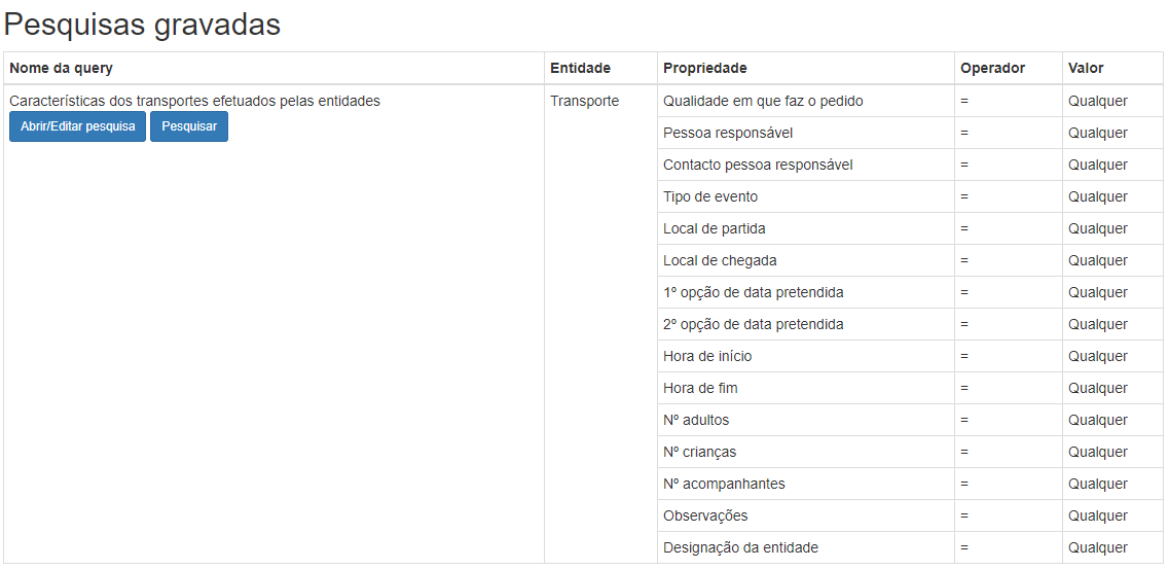


Figura 81 – Página Pesquisas Gravadas

Na página Pesquisas Gravadas é apresentada uma tabela com as pesquisas/*querys* gravadas e é possível observar em que tipo de entidade a pesquisa foi efetuada, quais as propriedades que foram selecionadas e respetivos valores e operadores introduzidos. Quando não é atribuído nenhum valor na pesquisa, decidiu-se apresentar a palavra "Qualquer".

Inferiormente ao nome da *query* podemos verificar dois botões. O primeiro "Abrir/Editar pesquisa", tal como o próprio nome indica, serve para abrir o formulário de pesquisa novamente, já com os dados preenchidos, e caso se pretenda, é possível editar essa pesquisa para obter novos resultados. Caso seja pretendido visualizar imediatamente os resultados da *query* guardada sem ser necessário apresentar o formulário de pesquisa, clica-se logo no segundo botão "Pesquisar" e somos direcionados para a página com os resultados.

5.6 Testes de usabilidade

Após a fase de implementação do sistema estar concluída foram efetuados testes de usabilidade. Os testes de usabilidade são realizados com utilizadores representativos do público alvo e consistem em avaliar um produto ou serviço, medindo a sua usabilidade ou facilidade de uso. Devido a estes testes é possível compreender melhor a relação do utilizador com o produto e identificar problemas na interação [42].

Para que os testes de usabilidade sejam executados eficazmente é necessário desenvolver um plano de teste sólido. Para isso é essencial definir o perfil dos utilizadores e recrutá-los, definir as tarefas para os utilizadores executarem, decidir que métricas serão avaliadas, e por fim analisar os resultados dos testes para proceder às alterações necessárias.

Segundo Jakob Nielsen, especialista em usabilidade, se o objetivo é expor os problemas de usabilidade do produto, num curto período de tempo, 4 a 5 participantes serão capazes de identificar a maioria (85%) dos problemas [43]. Portanto para a realização dos testes foram recrutados 5 utilizadores.

Antes de começar os testes é importante contextualizar os utilizadores. Deste modo foi proferida uma breve descrição sobre a plataforma e logo de seguida o utilizador explorou o sistema livremente durante cerca de dois minutos para se adaptar. Só após esta contextualização é que foi dado início aos testes. O guia onde se encontram os cenários e as tarefas a serem executadas pelos participantes podem ser consultados no anexo C.

Durante a realização dos testes, os participantes foram observados e foi-lhes pedido para que verbalizassem as suas dúvidas de modo a compreender melhor a ocorrência e a razão dos problemas. Para além disso, para cada participante foi registado o tempo gasto para completar as tarefas, e os números de erros cometidos em cada cenário. Os erros cometidos dizem respeito aos cliques que não contribuíram para completar as tarefas.

Entre os cinco utilizadores que participaram nos testes, três possuíam conhecimentos avançados na ótica do utilizador e dois possuíam conhecimentos médios.

Após a conclusão das tarefas os participantes responderam a um questionário de avaliação do sistema previamente definido cuja finalidade é a recolha de dados subjetivos sobre a sua satisfação. O questionário referido e respetivos resultados podem ser consultados no anexo D.

5.6.1 Resultados dos testes de usabilidade

Após a realização dos testes de usabilidade com os utilizadores, é fundamental analisar os dados resultantes, diagnosticar os problemas, e proceder às alterações necessárias para solucioná-los.

Os dados obtidos foram registados nas tabelas 6 e 7, em que na tabela 6 está representado o tempo que cada utilizador levou para completar as tarefas e na tabela 7 estão assinalados os números de erros que cada utilizador cometeu em cada cenário.

Cenário	Utilizador 1	Utilizador 2	Utilizador 3	Utilizador 4	Utilizador 5
1	4 min 23 s	6 min 15 s	3 min 25 s	4 min 20 s	5 min 03 s
2	1 min 56 s	2 min 23 s	3min 4 s	2 min 47 s	1 min 59 s
3	1 min 50 s	1 min 37 s	1min 53 s	1 min 43 s	1 min 34 s
4	58 s	1 min 39 s	1min 44 s	48 s	52 s
5	22 s	49 s	42 s	50 s	35 s
6	1 min 56 s	2 min 56 s	1 min 52 s	2 min 3 s	1 min 59 s
7	1 min 05 s	1 min 01 s	51 s	42 s	59 s
8	51 s	54 s	42 s	39 s	43 s
9	32 s	50 s	48 s	46 s	52 s

Tabela 6 - Resultados dos testes de usabilidade (tempos dos utilizadores)

Cenário	Utilizador 1	Utilizador 2	Utilizador 3	Utilizador 4	Utilizador 5
1	1	5	1	0	3
2	0	1	0	2	0
3	1	0	0	1	1
4	0	0	1	0	0
5	0	0	0	0	0
6	0	0	1	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	0	0

Tabela 7 - Resultados dos testes de usabilidade (erros dos utilizadores)

Como pode ser observado na tabela 7, houve uma maior ocorrência de erros nos primeiros três cenários.

No primeiro cenário, em que o objetivo era testar o *dashboard* do ponto de vista do munícipe, os utilizadores referiram que o *dashboard* não era muito intuitivo e sugeriram que deveria haver informação mais explícita de como proceder para efetuar os pedidos. Apesar de não ter ficado encarregue do desenvolvimento do *dashboard*, achou-se relevante realizar testes de usabilidade para ter uma noção de como os utilizadores se comportam a utilizar o sistema.

No segundo cenário, em que o objetivo era testar o componente Gestão de Propriedades, dois dos cliques incorretos ocorreram na pesquisa do tipo de entidade e o outro clique incorreto ocorreu na reordenação de propriedades, uma vez que o utilizador não associou que a reordenação era efetuada recorrendo ao *drag and drop*. Para corrigir estes problemas, foi decidido colocar um *placeholder* nos campos de pesquisa a informar do que deve ser introduzido, e no *modal* de reordenação de propriedades foi apenas adicionada uma pequena descrição de como a reordenação de propriedades deve ser efetuada.

Já no terceiro cenário, em que o objetivo era testar o *dashboard* da perspetiva de um funcionário da Câmara Municipal do Funchal, os utilizadores mencionaram que o modo como os dados introduzidos pelo munícipe eram apresentados poderia ser melhorado de forma a ficar mais legível.

Quanto aos restantes cenários, em que o objetivo era testar o componente Gestão de Pesquisas, a ocorrência de erros não foi significativa. No entanto, os utilizadores fizeram algumas observações, nomeadamente referiram que o botão de "Pesquisa" na página da Pesquisa Dinâmica devia ser colocado de forma a que ficasse mais visível e em relação às Pesquisas Gravadas um utilizador conseguiu identificar um *bug*. Esse *bug* surgia quando o utilizador clicava no botão "Abrir/Editar pesquisa" em que os valores escolhidos não ficavam pré-preenchidos. Posteriormente foram efetuadas correções para ambos os problemas.

Relativamente aos problemas encontrados no primeiro e terceiro cenários, estes dizem respeito ao *dashboard*, que aos poucos tem vindo a ser melhorado pelo colega que ficou responsável pelo seu desenvolvimento.

6 Conclusões e trabalho futuro

Muitos dos obstáculos que surgem nas organizações são devido à sua crescente complexidade. Na maior parte das vezes as dificuldades surgem porque não há uma distinção clara de responsabilidades e também porque as organizações não possuem ferramentas apropriadas para lidar com processos organizacionais complexos. Um dos motivos para a utilização da metodologia DEMO neste projeto foi, precisamente, para evitar que ocorressem problemas dessa natureza.

Após a realização deste projeto, pode-se afirmar que a metodologia DEMO revelou-se muito eficaz na modelação dos serviços, pois através das suas teorias, conceitos e diagramas obteve-se uma visão mais simplificada sobre os processos. Esta metodologia foca-se nos elementos essenciais, descartando informações irrelevantes e também fornece uma definição clara e completa de competências, autoridades e responsabilidades.

Em relação ao protótipo desenvolvido, este é totalmente dinâmico e adaptável, não só aos processos de uma organização específica, mas também a outras organizações com diferentes processos, o que o torna um sistema com grande potencialidade.

Embora os objetivos que nos foram propostos tenham sido alcançados, reconhecemos que o protótipo necessita de melhorias futuramente. O trabalho futuro passa, por exemplo, pela introdução do padrão completo das transações (dado que de momento só possui o padrão básico), pela automatização das transações de auto ativação (transações que ocorrem periodicamente numa determinada circunstância), pela implementação de um novo requisito para executar várias transações de uma vez, por criar um histórico de todos os dados, e ainda por melhorar alguns aspetos de design e de usabilidade. Após mais alguns desenvolvimentos, espera-se que o sistema esteja preparado para ser utilizado num contexto real com utilizadores reais.

As maiores dificuldades experienciadas surgiram logo no princípio do projeto devido à constante variação dos objetivos. Como consequência foi despendido uma grande quantidade de tempo e foram elaborados conteúdos que não possuíram muita relevância para o verdadeiro objetivo. Contudo, os objetivos acabaram por estabilizar e com isso surgiu a solução final apresentada neste relatório.

Quanto a nível pessoal, este projeto foi uma experiência muito enriquecedora. Para além de ter adquirido mais conhecimentos na área de Engenharia Organizacional, também tive a oportunidade de trabalhar com tecnologias interessantes, tais como, as *frameworks* Laravel e AngularJS, que anteriormente me eram desconhecidas.

Do meu ponto de vista, a realização deste projeto foi um ponto muito significativo do meu percurso académico, não só pela experiência obtida, mas também porque permitiu-me aplicar conhecimentos de várias áreas e desenvolver as minhas competências a nível de programação.

7 Referências

[1] J. Dietz, “Enterprise Ontology: Theory and Methodology”, Germany: Springer- Verlag Berlin Heidelberg, 2006.

[2] D. Aveiro, Engenharia Organizacional lectures, Universidade da Madeira, 2010.

[3] *Modeling an organization using Enterprise Ontology - by Johan Den Haan*. (2009). *The Enterprise Architect*. Retrieved 6 December 2017, from http://www.theenterprisearchitect.eu/blog/2009/10/10/modeling-an-organization-using-enterprise-ontology/

[4] Knowledge Discovery, Knowledge Engineering and Knowledge Management: Third International Joint Conference, IC3K 2011, Paris, France, October 26-29, 2011. Revised Selected Papers

[5] Johan den Hann, An Enterprise Ontology based approach to Model-Driven Engineering, Delft University of Technology Delft, 2009.

[6] Jaques, R. (2016). *O que é um Framework? Para que serve? - PHPit*. *PHPit - Blog de PHP do Rafa Jaques*. Retrieved 29 September 2018, from http://www.phpit.com.br/artigos/o-que-e-um-framework.phpit

[7] *Node.js*. (2017). *En.wikipedia.org*. Retrieved 2 October 2017, from https://en.wikipedia.org/wiki/Node.js

[8] Chrome V8 | Google Developers. (2017). Google Developers. Retrieved 2 October 2017, from https://developers.google.com/v8/

[9] Foundation, N. (2017). About | Node.js. Node.js. Retrieved 2 October 2017, from https://nodejs.org/en/about/

[10] *01 - What is npm? | npm Documentation*. (2018). *Docs.npmjs.com*. Retrieved 18 July 2018, from https://docs.npmjs.com/getting-started/what-is-npm

[11] *Node.js NPM*. (2018). *W3schools.com*. Retrieved 18 July 2018, from https://www.w3schools.com/nodejs/nodejs_npm.asp

[12] Express - Node.js web application framework. (2017). Expressjs.com. Retrieved 2 October 2017, from http://expressjs.com/

[13] Node.js Express Framework. (2017). www.tutorialspoint.com. Retrieved 2 October 2017, from https://www.tutorialspoint.com/nodejs/nodejs_express_framework.htm

[14] *Advantages of using Express js | Impressico Business Solutions*. (2017). *Impressico.com*. Retrieved 6 October 2017, from http://www.impressico.com/advantages-of-using-express-js/

[15] Solutions. (2017). *Benefits of Express JS . Slideshare.net*. Retrieved 3 October 2017, from https://www.slideshare.net/Techticsolution/benefits-of-express-js

[16] *Meteor.js: Construindo aplicações web com Node.js e MongoDB*. (2017). *DevMedia*. Retrieved 7 October 2017, from https://www.devmedia.com.br/meteor-js-construindo-aplicacoes-web-com-node-js-e-mongodb/32953

[17] *Plataforma Meteor, baseada em JavaScript, chega a versão 1.0 e ganha popularidade entre desenvolvedores*. (2014). *EXAME*. Retrieved 7 October 2017, from https://exame.abril.com.br/tecnologia/plataforma-meteor-baseada-em-javascript-chega-a-versao-1-0-e-ganha-popularidade-entre-desenvolvedore/

[18] *Meteor Js – ReviewStories*. (2017). *ReviewStories*. Retrieved 7 October 2017, from https://thereviewstories.com/meteor-js/

[19] *CodeIgniter Web Framework*. (2017). *Codeigniter.com*. Retrieved 9 October 2017, from https://codeigniter.com/

[20] Joshi, Y. (2011). *What is Code Igniter and what are its advantages and disadvantages?*. *Esds.co.in*. Retrieved 9 October 2017, from https://www.esds.co.in/blog/what-is-code-igniter-and-what-are-its-advantages-and-disadvantages/#sthash.7qSYk60h.x0iWEv6G.dpbs

[21] Otwell, T. (2018). *Introduction - Laravel - The PHP Framework For Web Artisans*. *Laravel.com*. Retrieved 9 October 2017, from https://laravel.com/docs/4.2/introduction

[22] *Porque e como começar a utilizar o Laravel*. (2017). *PET Sistemas de Informação*. Retrieved 9 October 2017, from http://coral.ufsm.br/pet-si/index.php/porque-e-como-comecar-a-utilizar-o-laravel/

[23] *CodeIgniter X Laravel - Prós e contras de cada framework | RBtech Developer*. (2016). *RBtech Developer*. Retrieved 3 November 2017, from http://dev.rbtech.info/codeigniter-vs-laravel-melhor/

[24] *package.json | npm Documentation*. (2017). *Docs.npmjs.com*. Retrieved 19 October 2017, from https://docs.npmjs.com/files/package.json

[25] *Best practices for Express app structure*. (2014). *Terlici*. Retrieved 31 October 2017, from https://www.terlici.com/2014/08/25/best-practices-express-structure.html

[26] *Composer*. (2017). *Getcomposer.org*. Retrieved 21 October 2017, from https://getcomposer.org/

[27] Otwell, T. (2017). *Directory Structure - Laravel - The PHP Framework For Web Artisans*. *Laravel.com*. Retrieved 21 October 2017, from https://laravel.com/docs/5.4/structure

[28] Otwell, T. (2017). *Database: Migrations - Laravel - The PHP Framework For Web Artisans*. *Laravel.com*. Retrieved 24 October 2017, from https://laravel.com/docs/5.4/migrations

[29] Otwell, T. (2017). *Eloquent: Getting Started - Laravel - The PHP Framework For Web Artisans*. *Laravel.com*. Retrieved 24 October 2017, from https://laravel.com/docs/5.4/eloquent

[30] *Requisito*. (2017). *Pt.wikipedia.org*. Retrieved 19 November 2017, from https://pt.wikipedia.org/wiki/Requisito

[31] Leite, J., Leite, J., & completo, V. (2007). *Requisitos de Software*. *Engenhariadesoftware.blogspot.com*. Retrieved 19 November 2017, from http://engenhariadesoftware.blogspot.com/2007/05/requisitos-de-software.html

[32] *Requisitos Não Funcionais e Arquitetura de Software*. (2017). *DevMedia*. Retrieved 20 November 2017, from https://www.devmedia.com.br/artigo-engenharia-de-software-3-requisitos-nao-funcionais/9525

[33] *Arquitetura de Software Na Pratica*. (2017). *Pt.slideshare.net*. Retrieved 14 June 2018, from https://pt.slideshare.net/kieras/arquitetura-de-software-na-prtica-1476447

[34] *Client–server model*. (2017). *En.wikipedia.org*. Retrieved 14 June 2018, from https://en.wikipedia.org/wiki/Client%E2%80%93server_model

[35] *AngularJS*. (2018). *Docs.angularjs.org*. Retrieved 1 May 2018, from https://docs.angularjs.org/guide/introduction

[36] *Single-page application*. (2018). *En.wikipedia.org*. Retrieved 1 May 2018, from https://en.wikipedia.org/wiki/Single-page_application

[37] *O que é AngularJS - Portal GSTI*. (2018). *Portal GSTI*. Retrieved 4 May 2018, from https://www.portalgsti.com.br/angularjs/sobre/

[38] Mark Otto, a. (2018). *Bootstrap*. *Getbootstrap.com*. Retrieved 6 May 2018, from https://getbootstrap.com/

[39] *7 razões para desenvolver seus web designs no Bootstrap - iMasters - We are Developers*. (2017). *iMasters - We are Developers*. Retrieved 6 May 2018, from https://imasters.com.br/desenvolvimento/7-razoes-para-desenvolver-seus-web-designs-no-bootstrap

[40] Leone, L., Soares, D., Lug, A., & Soares, D. (2017). *Bootstrap: o que é, porque usar e como começar com o framework*. *Becode*. Retrieved 8 May 2018, from https://becode.com.br/bootstrap-o-que-e-porque-usar-e-como-comecar/

[41] Welcome to PhpSpreadsheet's documentation - PhpSpreadsheet Documentation. (2018). Phpspreadsheet.readthedocs.io. Retrieved 21 May 2018, from https://phpspreadsheet.readthedocs.io/en/develop/

[42] Affairs, A. (2018). *Usability Testing | Usability.gov*. *Usability.gov*. Retrieved 14 April 2018, from https://www.usability.gov/how-to-and-tools/methods/usability-testing.html

[43] *Testes de Usabilidade*. (2018). *Pt.slideshare.net*. Retrieved 14 April 2018, from https://pt.slideshare.net/DanieleZ/testes-de-usabilidade-77273046

[44] *Balsamiq Mockups | Download*. (2018). *TechTudo*. Retrieved 11 February 2017, from http://www.techtudo.com.br/tudo-sobre/balsamiq-mockups.html

[45] *Prototipação, importância no desenvolvimento de software*. (2013). *Desenvolvimento de software sob medida*. Retrieved 11 February 2017, from http://dextra.com.br/pt/blog/prototipacao-e-sua-importancia-no-desenvolvimento-de-software/

[46] *WordPress*. (2018). *Pt.wikipedia.org*. Retrieved 22 April 2017, from https://pt.wikipedia.org/wiki/WordPress

[47] *Entendendo o MVC (Model-View-Controller)*. (2013). *DigitalDev*. Retrieved 22 April 2017, from http://www.digitaldev.com.br/2013/01/18/entendendo-o-mvc-model-view-controller/

[48] Bastos, D. (2011). *O que é Model-view-controller (MVC)?*. *Oficina da Net*. Retrieved 22 April 2017, from https://www.oficinadanet.com.br/artigo/desenvolvimento/o_que_e_model-view-controller_mvc

8 Anexos

8.1 Anexo A – Trabalho complementar

O objetivo inicial para este projeto seria o desenvolvimento de uma nova plataforma *online* que incluía todos os requerimentos existentes na antiga plataforma da Câmara Municipal do Funchal, mas em formato digital.

Nessa nova plataforma o munícipe teria ao seu dispor a lista de requerimentos referentes aos diversos serviços da Autarquia e poderia preenchê-los e submetê-los *online*.

Um dos objetivos da implementação dessa nova plataforma seria evitar que o munícipe tivesse de realizar o *download* do requerimento em PDF (*Portable Document Format*) para proceder à sua impressão, de modo a preenchê-lo manualmente, e ainda evitar que os munícipes se deslocassem pessoalmente até aos serviços de atendimento da Câmara Municipal do Funchal para entregar os seus pedidos. Assim, os munícipes poderiam efetuar os seus pedidos na comodidade de sua casa, ou através de um dispositivo móvel, contribuindo também para a diminuição do papel.

Com o intuito de implementar essa nova plataforma, primeiramente foi necessário apurar quais eram os requerimentos mais requisitados pelos munícipes. Segundo as informações concedidas pela Câmara Municipal do Funchal, os mais requisitados eram: cedência de transporte, marcação de audiência, averbamento da licença de abertura, colocação de andaime/tapume/abertura de valas, atribuição de licença de venda ambulante, emissão e renovação do cartão de venda ambulante, e atribuição de tarifa familiar e social.

De modo a compreender melhor como se processa cada pedido, foram elaboradas algumas questões à Câmara Municipal do Funchal, e seguidamente foi realizado uma breve descrição referente a cada pedido. Essa descrição pode ser consultada na secção 8.1.1.

8.1.1 Descrição de Pedidos

8.1.1.1 Cedência de transporte

O pedido de cedência de transporte não pode ser efetuado por um munícipe em particular, mas sim por entidades sejam elas culturais, desportivas, recreativas, juvenis, sociais, religiosas ou de ensino.

Os pedidos de cedência de transporte devem ser feitos através do requerimento próprio que está disponível no portal de Serviços Online da Câmara na área dos requerimentos. O pedido para este serviço pode ser entregue presencialmente ou submetido *online*.

Os pedidos de cedência deverão conter a identificação completa e morada da entidade requerente, o tipo de evento, o objetivo da deslocação, número de pessoas a deslocar, identificação de um responsável pela deslocação e seu respetivo contacto, dia

hora e local da partida e hora previsível de chegada. Em caso de transporte de crianças em coletivo é necessária uma declaração de responsabilidade.

A decisão de cedência de transporte baseia-se na disponibilidade de viaturas para a data e hora pretendida. Posteriormente os serviços da Câmara Municipal confirmarão as cedências ou informarão a sua impossibilidade através do contacto telefónico disponibilizado pela entidade requerente.

8.1.1.2 Marcação de Audiência

O formulário apropriado para este serviço encontra-se disponível no portal de Serviços Online da Câmara na área dos requerimentos, e tem necessariamente de ser submetido *online*.

Este formulário permite efetuar um pedido de marcação de audiência com o Senhor Presidente da Câmara, Vice-Presidente ou Vereador. Para que esse pedido seja aceite é obrigatório indicar a identificação do requerente, assim como a sua morada completa, os seus contactos (telemóvel e email) e ainda indicar o assunto da audiência. Opcionalmente, ainda poderá ser indicado uma ou mais referências de documentos que estejam associados ao assunto indicado.

Posteriormente o cidadão/munícipe será informado do dia e da hora da audiência, através de um dos meios de comunicação indicados.

8.1.1.3 Averbamento da licença de abertura

O pedido de averbamento de licença de abertura, apenas pode ser entregue presencialmente e deve fazer-se acompanhar de outros documentos, nomeadamente da fotocópia do cartão de contribuinte, da fotocópia e original da escritura de trespasse/arrendamento/cessão de exploração e fotocópia e original da escritura da sociedade.

O pedido de averbamento de licença de abertura deve conter a identificação completa e morada do munícipe requerente, o nome e o número do estabelecimento adquirido, o sítio e freguesia do mesmo.

Para que este serviço seja aceite é necessário efetuar o pagamento de taxas que estão fixadas nos termos do Regulamento da Tabela de Taxas e Licenças Municipais.

8.1.1.4 Colocação de andaime/tapume ou abertura de valas

O formulário para estes pedidos são idênticos e visam requerer a licença de colocação de andaime/tapume ou abertura de valas na via pública, e apenas pode ser entregue presencialmente. Sendo assim é obrigatório conter a identificação completa e morada do munícipe.

Este formulário deverá conter a identificação da rua e da freguesia onde o andaime/tapume irá ser colocado ou onde a vala será aberta, por quantos dias, a extensão ocupada e com que finalidade. Para que este pedido seja aceite, também é necessário entregar uma planta do local, alvará de licença de construção e pagar as taxas devidas referentes ao pedido.

8.1.1.5 Atribuição de Licença de Venda Ambulante

Este formulário encontra-se disponível no portal de Serviços Online na área dos formulários na secção de Fiscalização Municipal e permite obter a licença para exercer a atividade de venda ambulante.

Os vendedores ambulantes só podem exercer a sua atividade no concelho do Funchal se forem titulares de licença emitida pelo município e portadores de cartão de vendedor ambulante válido.

O vendedor é que tem de especificar qual é período em que pretende a licença de venda ambulante e a licença é atribuída normalmente para o local definido pelo vendedor, a não ser que seja num local totalmente inadequado que impeça ou de qualquer forma dificulte o trânsito.

Posteriormente, a licença será emitida mediante o pagamento da taxa em vigor.

Este pedido apenas pode ser entregue presencialmente e deve fazer-se acompanhar da fotocópia do cartão de contribuinte, do bilhete de identidade, do cartão de vendedor ambulante e de uma declaração de autorização do proprietário do terreno se a venda for realizada em local particular.

O pedido de atribuição de licença de venda ambulante deve conter a identificação da área total (em metros quadrados), do sítio, freguesia e tipo de área (terreno particular/via pública) em que a venda será realizada. Ainda deverá ser especificado por quanto tempo o vendedor pretende permanecer com a licença e por que ocasião.

8.1.1.6 Emissão de Cartão de Venda Ambulante

O cartão de Venda Ambulante é obrigatório para os vendedores poderem exercer a sua atividade, é apenas válido no Funchal e pelo período de um ano a contar da data da sua emissão ou renovação.

O formulário para este serviço encontra-se disponível no portal de Serviços Online na área dos formulários na secção de Fiscalização Municipal e apenas pode ser entregue presencialmente, pelo que deverá conter a identificação completa e morada do requerente.

Adicionalmente ao pedido, o requerente deve apresentar duas fotografias, cartão de contribuinte fiscal, bilhete de identidade e documento comprovativo de inscrição nas Finanças ou fotocópia da declaração de IRS (Imposto sobre o Rendimento de Pessoas Singulares).

8.1.1.7 Renovação de Cartão de Venda Ambulante

Após um ano, caso os cidadãos estejam interessados em continuar a exercer a sua atividade, é-lhes permitido efetuar um pedido de renovação de cartão.

O formulário para este pedido pode ser encontrado na área dos formulários no portal de Serviços Online e apenas pode ser entregue presencialmente.

O pedido de renovação deve conter a identificação completa e morada do requerente e o número do antigo cartão de vendedor ambulante. Para que este pedido seja aceite é obrigatório efetuar o pagamento das taxas relativas à renovação.

8.1.1.8 Atribuição de tarifa familiar

A tarifa familiar da água é um tarifário especialmente destinado a agregados familiares com 5 ou mais pessoas.

O formulário para este pedido só pode ser entregue presencialmente e está disponível no portal de Serviços Online na área dos formulários na secção de Águas e Saneamento.

Neste formulário deverá estar indicado o número de contrato, o consumidor, a instalação e ainda deverá ser especificado os dados de todos os membros do agregado familiar, nomeadamente o nome, o grau de parentesco, a data de nascimento e o número de contribuinte.

Ainda deverão ser apresentados uma cópia da última declaração de IRS, para confirmação do agregado familiar, e um atestado da junta de freguesia para confirmação de residência fixa e permanente de todos os elementos do agregado na mesma morada.

8.1.1.9 Atribuição de Tarifa Social

A tarifa social foi criada a pensar nas famílias mais carenciadas, proporcionando-lhes assim descontos na fatura da água.

O formulário para este pedido só pode ser entregue presencialmente e está disponível no portal de Serviços Online na área dos formulários na secção de Águas e Saneamento.

Neste formulário deverá estar indicado o número de contrato, o consumidor, a instalação e ainda deverá ser especificado os dados de todos os membros do agregado familiar, nomeadamente o nome, o grau de parentesco, a data de nascimento e o número de contribuinte.

Ainda deverá ser apresentado um atestado da junta de freguesia para confirmação do agregado familiar e cópia da declaração do rendimento ou da pensão que recebe.

8.1.2 TRT

Em complementação à descrição, foi efetuada a tabela 8 onde estão identificadas as transações e respetivo tipo de resultado. Nesta tabela ainda é possível visualizar quem inicia e quem executa cada transação.

ID	Transação	Tipo de Resultado	Papel Iniciador	Agente Iniciador	Papel Executor	Agente Executor
T1	Cedência de Transporte	[Cedência de transporte] foi efetuado	Requerente de Transporte	Entidade	Atribuidor de Transporte	Chefe da divisão de gestão de frota

T2	Marcação de audiência	Marcação de [audiência] foi efetuada	Requerente de marcação de audiência	Munícipe	Marcador de audiência	Chefe da divisão de atendimento e administração
T3	Averbamento da licença de abertura	Averbamento da [licença de abertura] foi efetuado	Requerente de averbamento da licença de abertura	Munícipe	Realizador de averbamento da licença de abertura	Chefe do departamento financeiro
T4	Pagamento das taxas referentes à licença de abertura	Pagamento das taxas referentes à [licença de abertura] foi efetuado		Administrativo	Pagador da taxa referente ao averbamento de licença de abertura	Munícipe
T5	Decisão sobre proposta de colocação de andaime	Decisão sobre proposta de colocação de [andaime] foi efetuado	Requerente de colocação de andaime	Munícipe	Decisor sobre proposta de colocação de andaime	Chefe da divisão de mobilidade e trânsito
T6	Pagamento de taxas relativas à colocação de andaime	Pagamento das taxas relativas à colocação de [andaime] foi efetuado		Administrativo	Pagador de taxas relativas à colocação de andaime	Munícipe
T7	Atribuição de licença de venda ambulante	Atribuição de [licença de venda ambulante] foi efetuado	Requerente de Licença	Munícipe	Atribuidor de Licença	Chefe da divisão de fiscalização municipal
T8	Pagamento referente à atribuição de licença de venda ambulante	Pagamento referente à atribuição de [licença de venda		Administrativo	Pagador de atribuição de licença de venda ambulante	Munícipe
T9	Emissão de cartão de venda ambulante	Emissão do [cartão de venda ambulante] foi efetuado	Requerente de Emissão de Cartão	Munícipe	Emissor de Cartão	Chefe da divisão de fiscalização municipal
T10	Pagamento referente à emissão do cartão de	Pagamento referente à emissão do [cartão de venda		Administrativo	Pagador do cartão de venda ambulante	Munícipe

	venda ambulante	ambulante] foi efetuado				
T11	Renovação de cartão de venda ambulante	[Renovação de cartão de venda ambulante] foi efetuado	Requerente de Renovação de Cartão	Munícipe	Renovador de Cartão	Chefe da divisão de fiscalização municipal
T12	Atribuição de Tarifa Familiar	Atribuição de [Tarifa Familiar] foi efetuado	Requerente de Tarifa Familiar	Munícipe	Atribuidor de Tarifa Familiar	Chefe da divisão de águas
T13	Atribuição de Tarifa Social	Atribuição de [Tarifa Social] foi efetuado	Requerente de Tarifa Social	Munícipe	Atribuidor de Tarifa Social	Chefe da divisão de águas

Tabela 8 – TRT pedidos mais solicitados

8.1.3 OFD

De seguida, na figura 80, é apresentado o OFD que foi construído através do TRT, apresentado anteriormente.

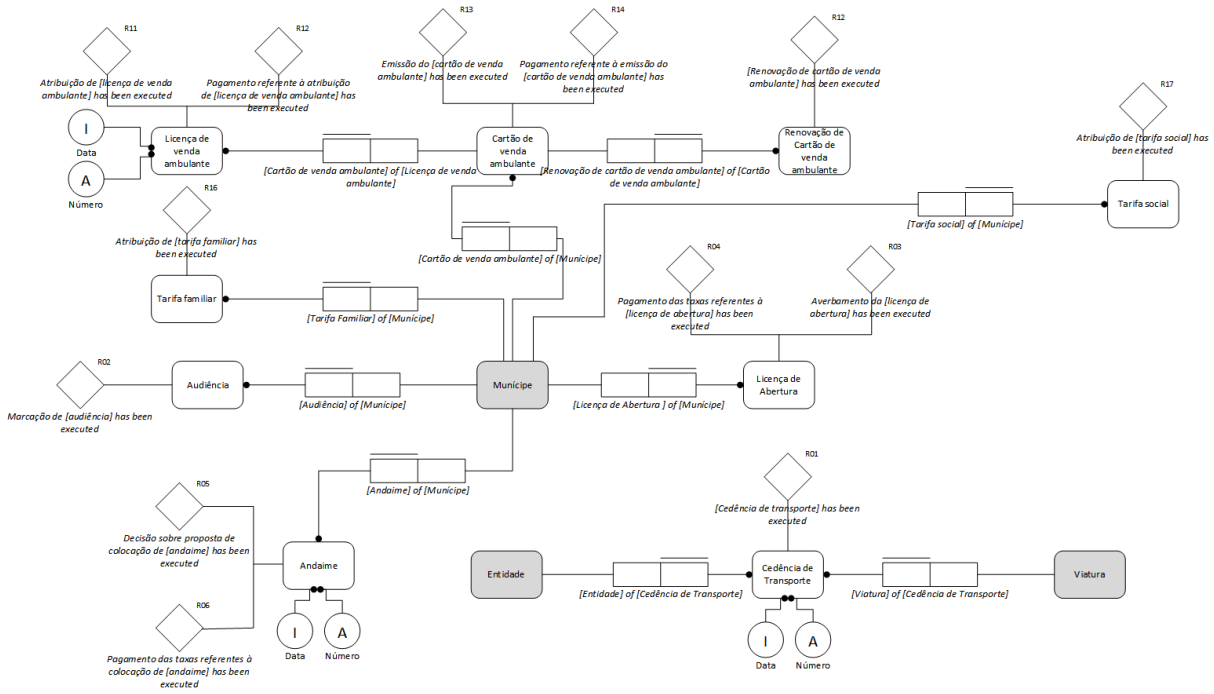


Figura 82- OFD pedidos mais solicitados

8.1.4 Protótipos

Como inicialmente estava prevista a implementação da nova plataforma, foram criados protótipos recorrendo ao software Balsamiq Mockups. Este programa oferece um meio termo entre os rascunhos de baixa e alta fidelidade, gerando protótipos com aparência de desenho [44].

Os protótipos são representações visuais do produto que está sendo desenvolvido e têm como objetivo facilitar o entendimento dos requisitos, apresentar conceitos e funcionalidades do software [45].

Inicialmente foram realizados protótipos para o lado do utilizador e posteriormente os protótipos para o lado do administrador, que se encontram, respetivamente, na secção 8.1.4.1 e 8.1.4.2.

É de referir que o desenvolvimento dos protótipos foi interrompido quando nos foi avisado que a plataforma já estava a ser desenvolvida por uma empresa em particular.

8.1.4.1 Protótipos do lado do utilizador

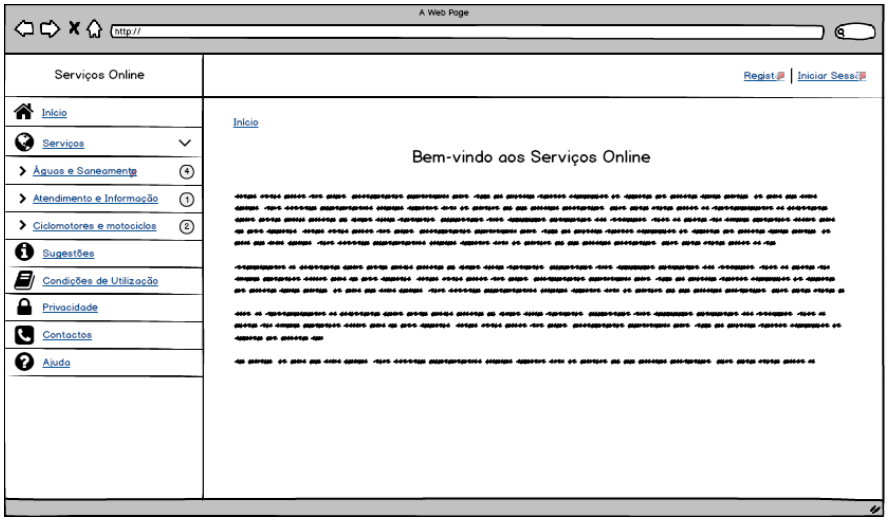


Figura 83 - Página Inicial (antes do login)

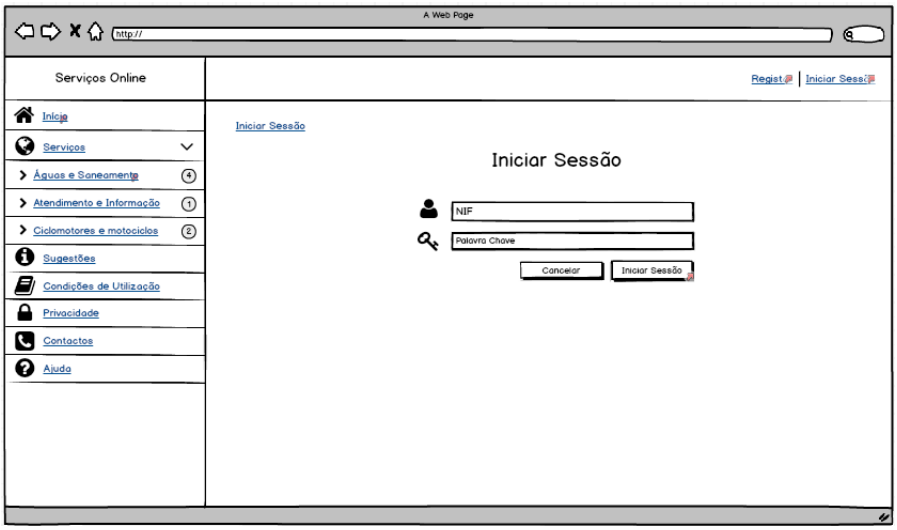


Figura 84 - Página para iniciar sessão

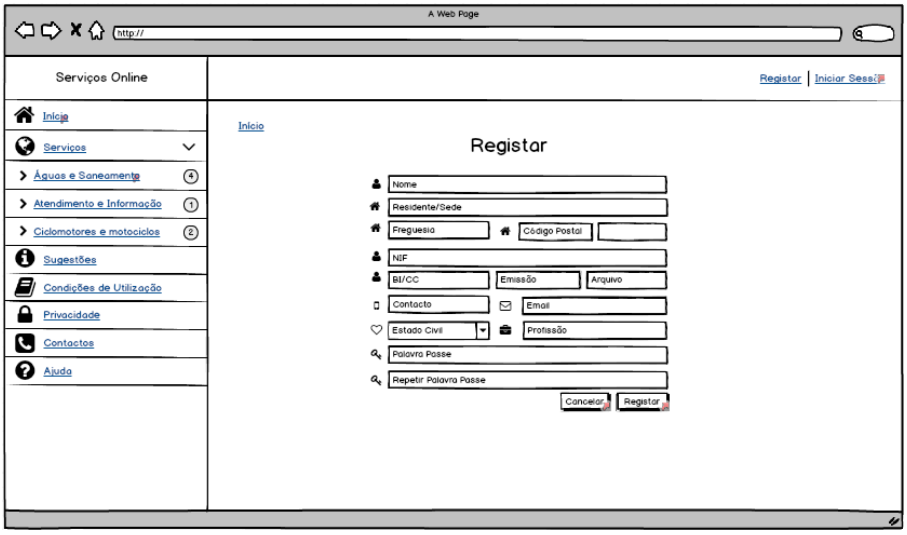


Figura 85 - Página para registar utilizadores

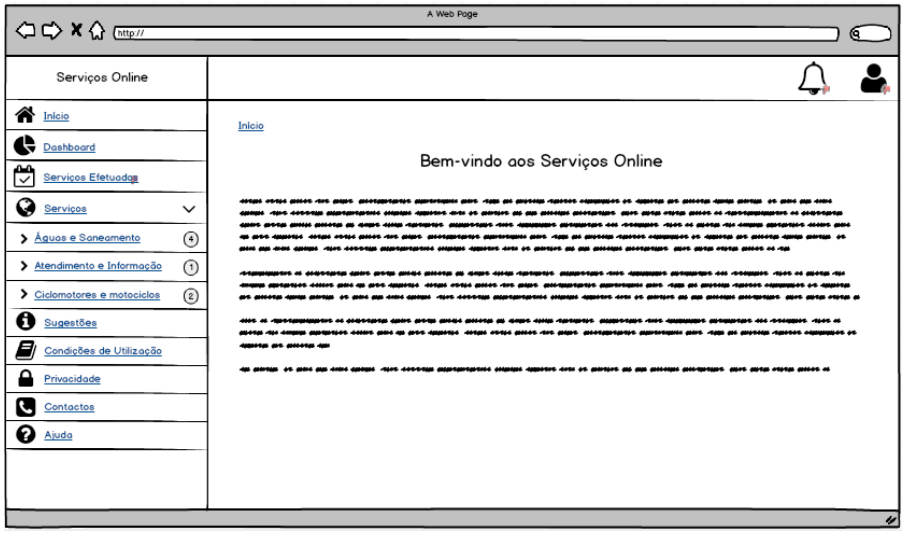


Figura 86 - Página inicial utilizador

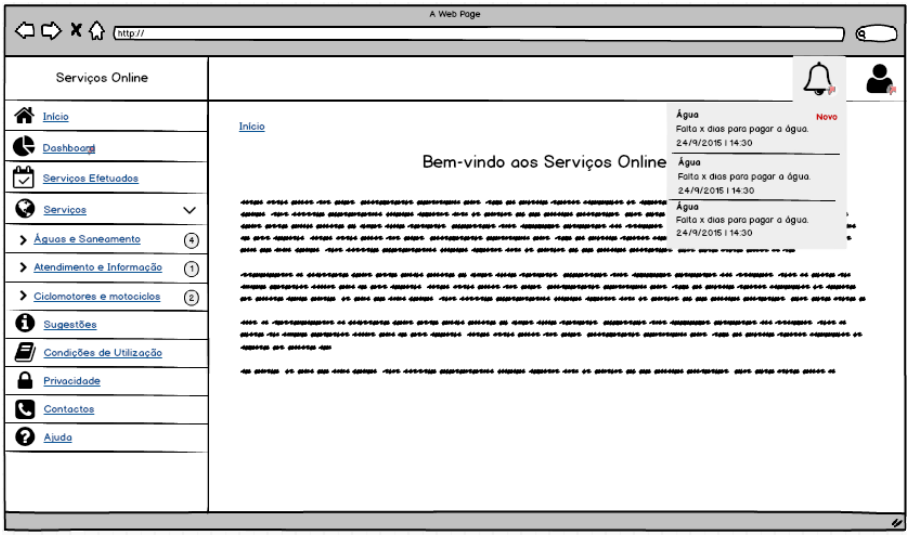


Figura 87 - Página inicial utilizador (notificações)

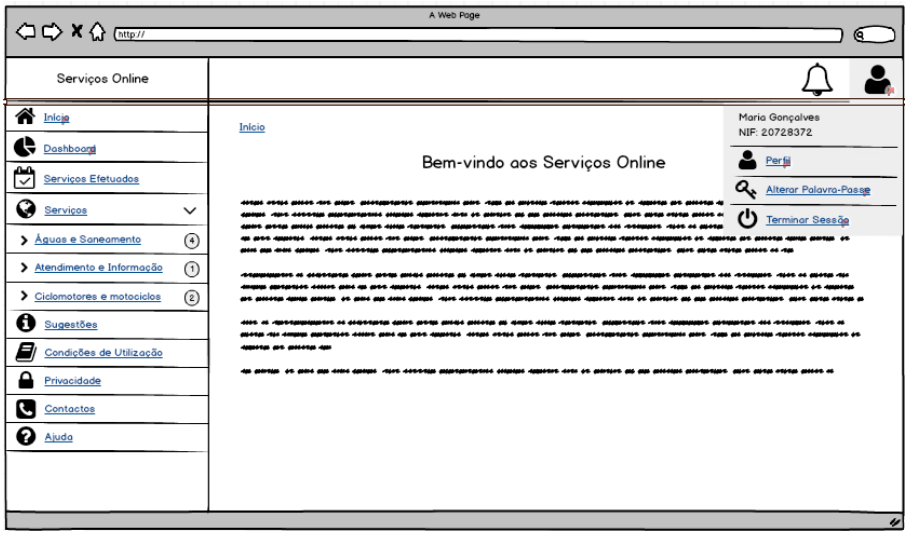


Figura 88 - Página inicial utilizador (dados pessoais)

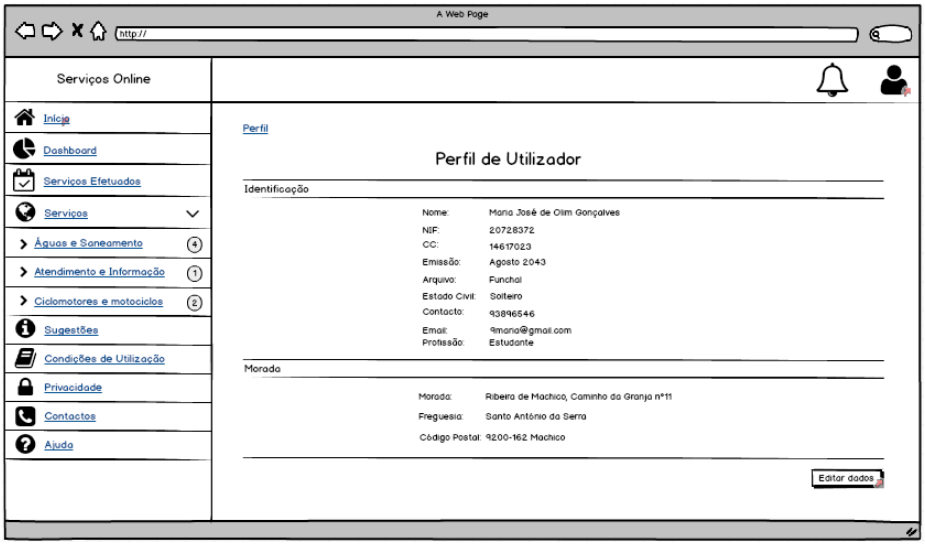


Figura 89 - Página perfil de utilizador

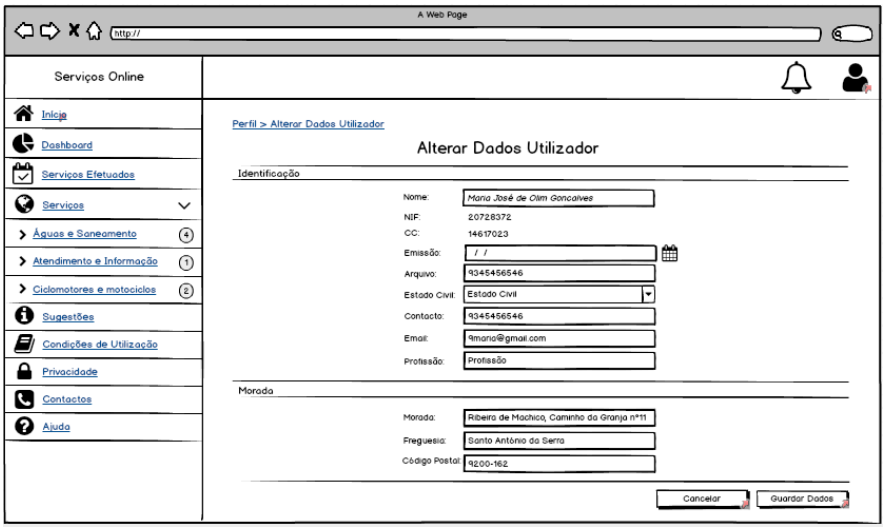


Figura 90 - Página para alterar dados de utilizador

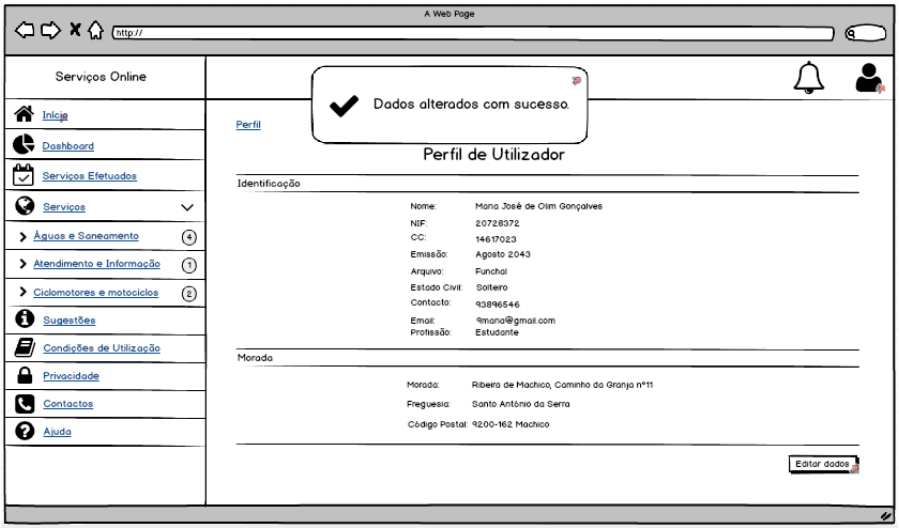


Figura 91 - Notificação dados alterados com sucesso

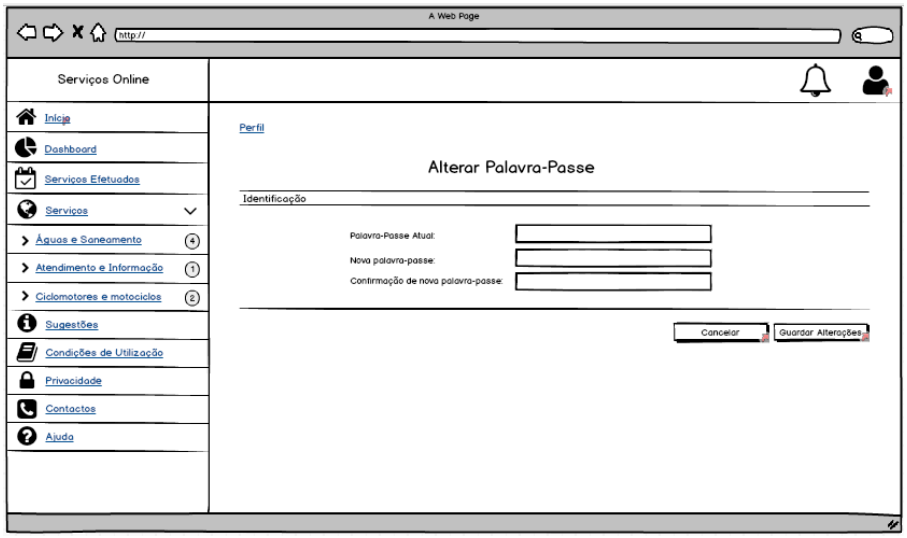


Figura 92 - Página alterar palavra-passe

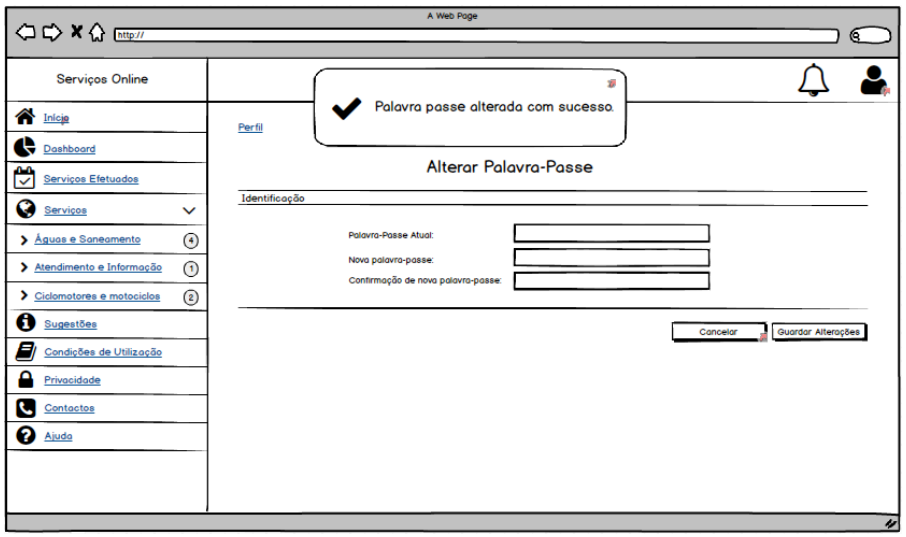


Figura 93 - Notificação palavra-passe alterada com sucesso

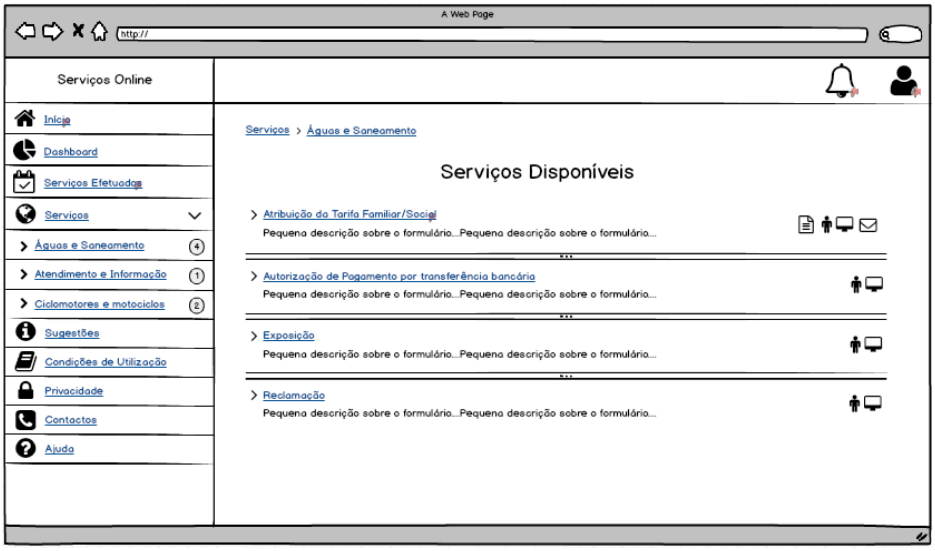


Figura 94 - Página Serviços Disponíveis

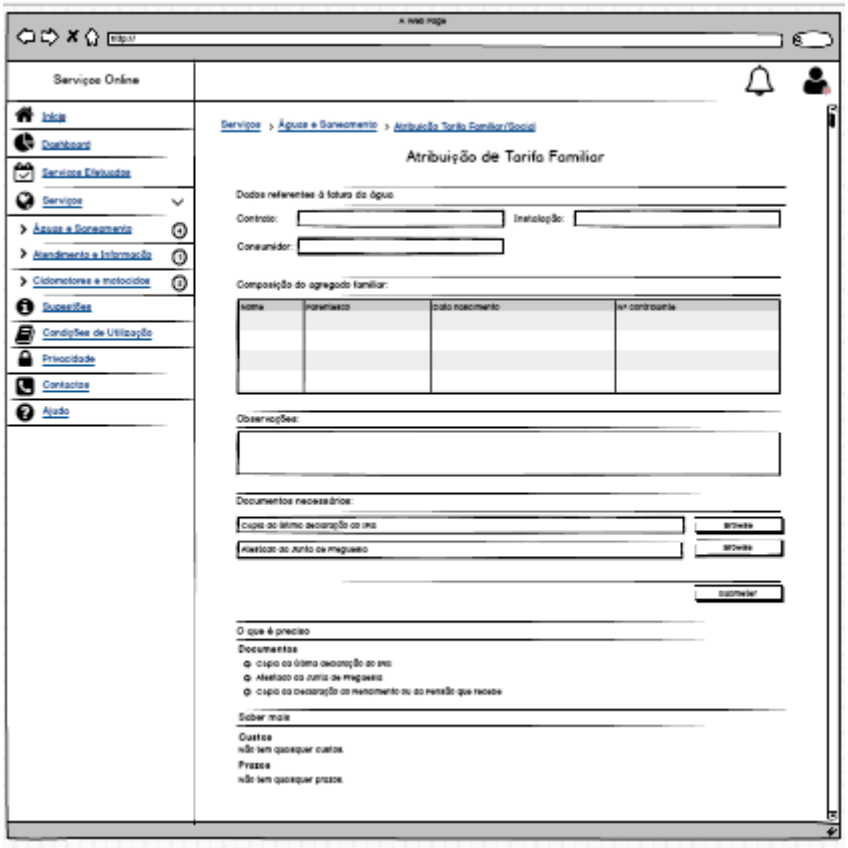


Figura 95 - Página com formulário de atribuição de tarifa familiar

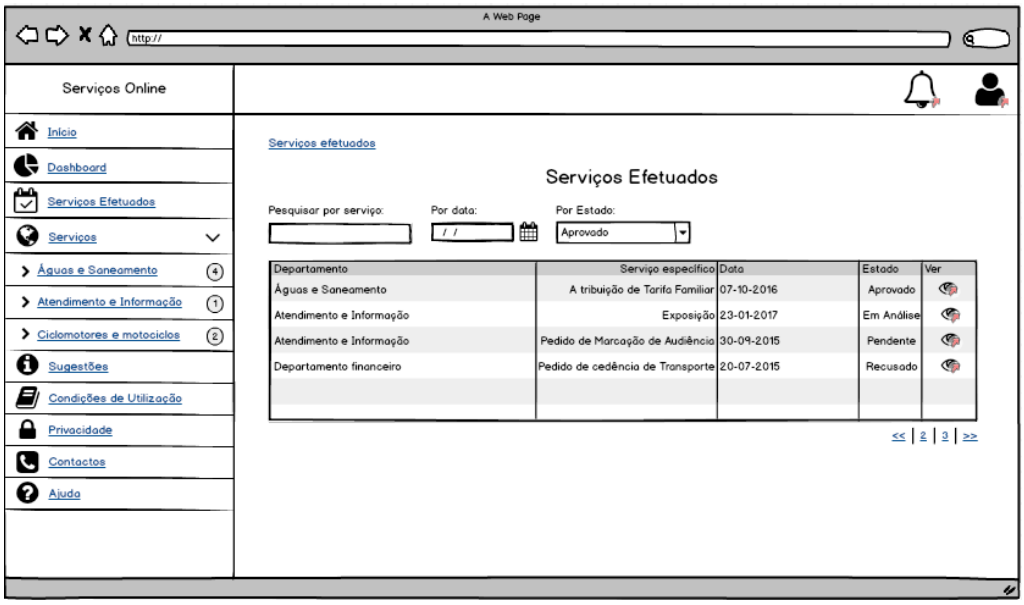


Figura 96 - Página serviços efetuados

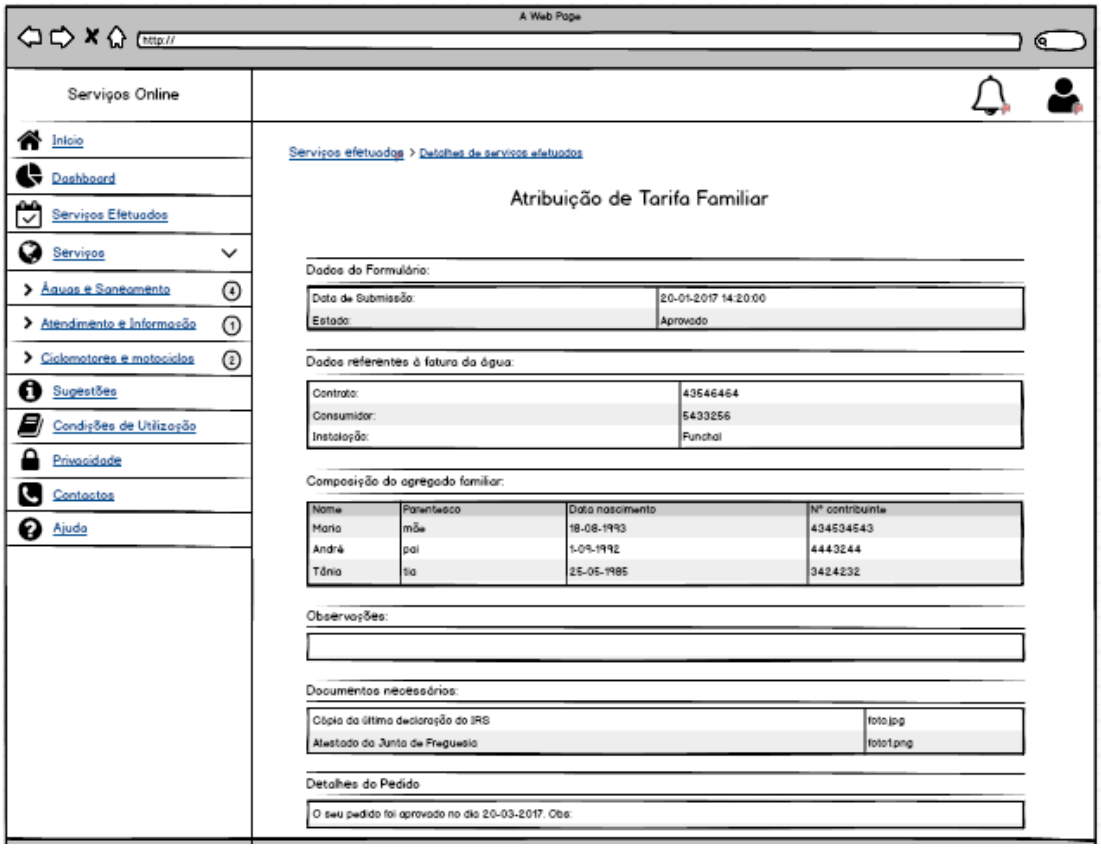


Figura 97 - Página para visualizar detalhes de serviços efetuados com estado aprovado

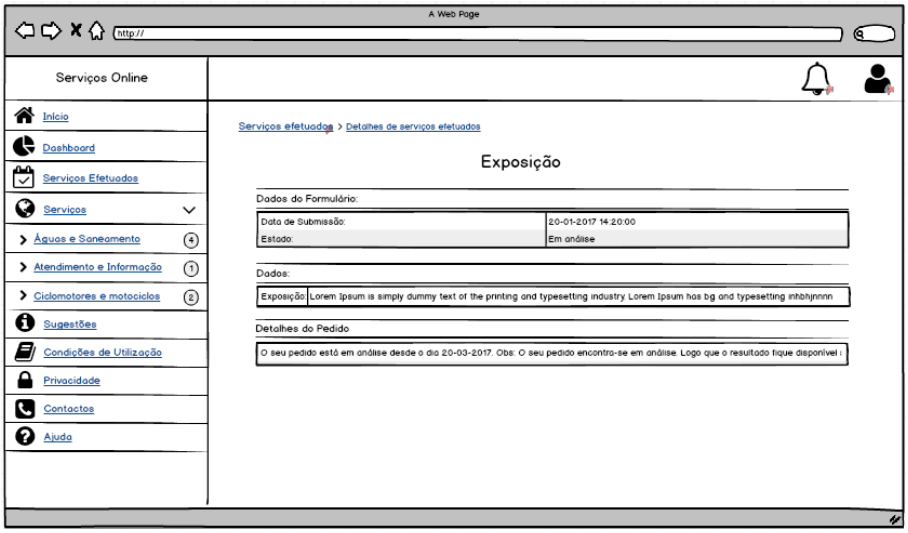


Figura 98 - Página para visualizar detalhes de serviços efetuados com estado em análise

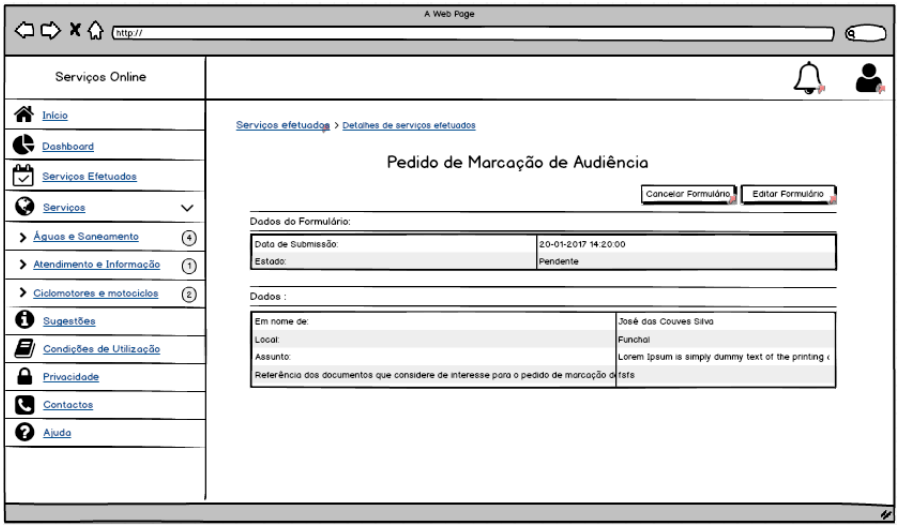


Figura 99- Página para visualizar detalhes de serviços efetuados com estado pendente

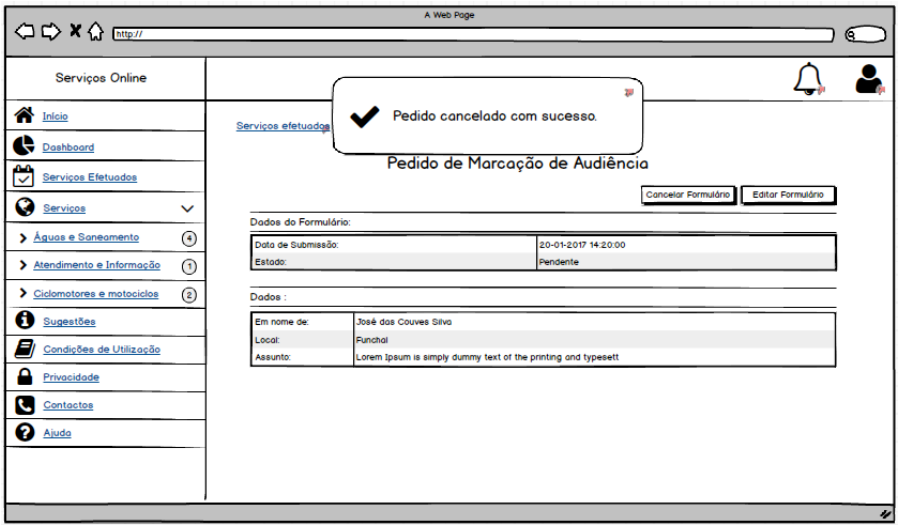


Figura 100 - Notificação pedido cancelado com sucesso

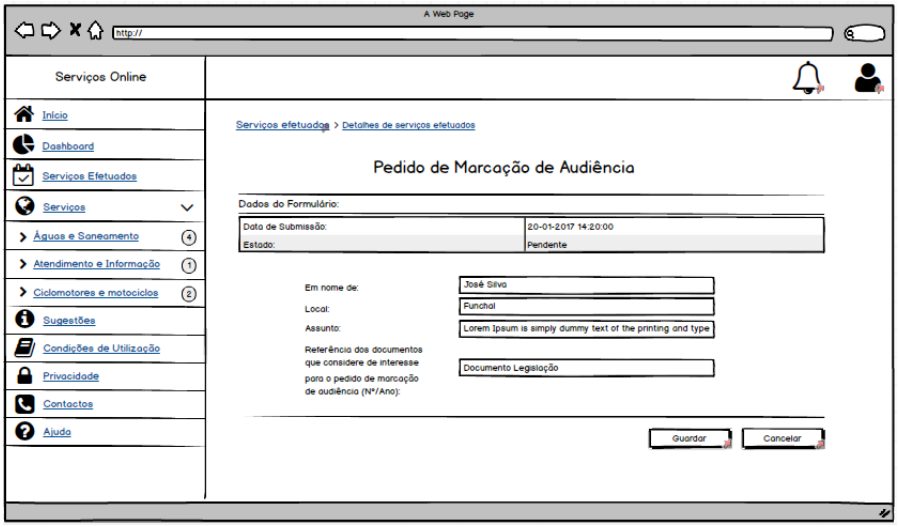


Figura 101 - Página editar detalhes de pedidos com estado pendente

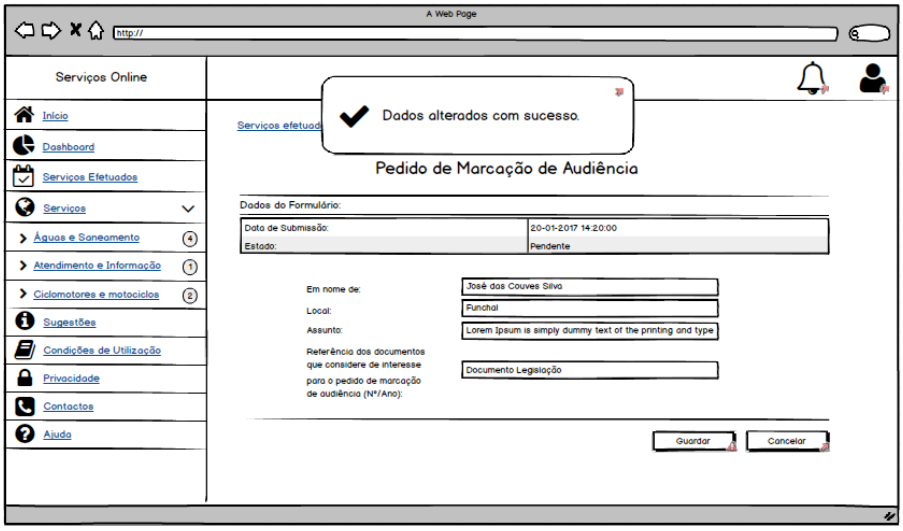


Figura 102 - Notificação dados editados com sucesso

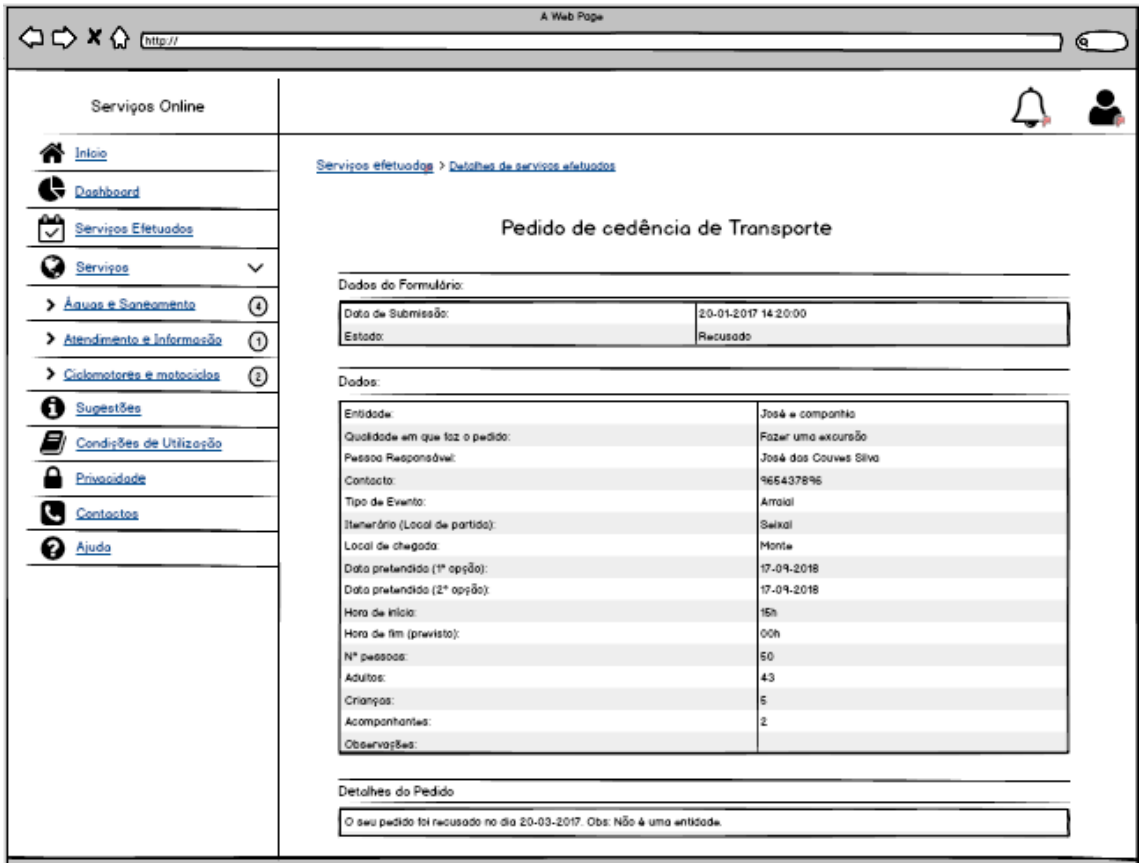


Figura 103- Página para visualizar detalhes de serviços efetuados com estado recusado

8.1.4.2 Protótipo do lado do administrador

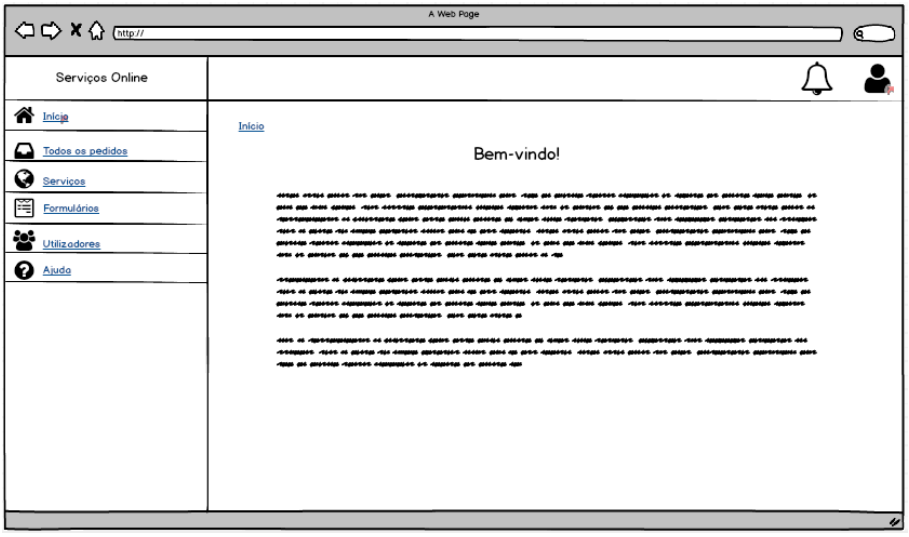


Figura 104 - Página inicial Administrador

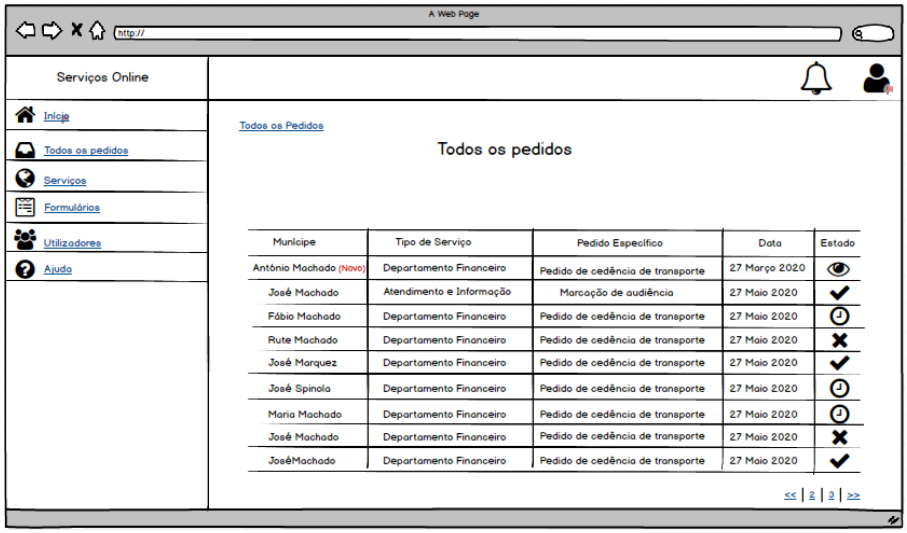


Figura 105 - Página todos os pedidos efetuados pelos munícipes

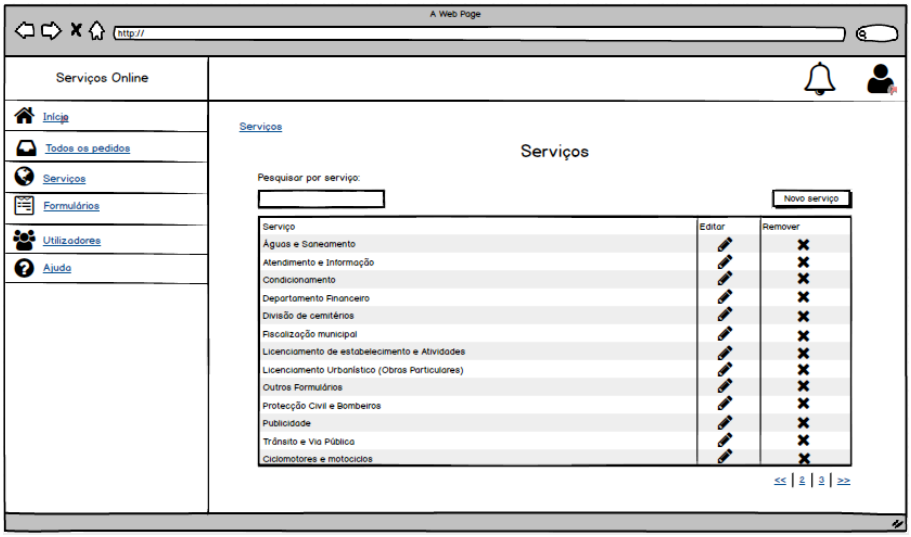


Figura 106 - Página serviços

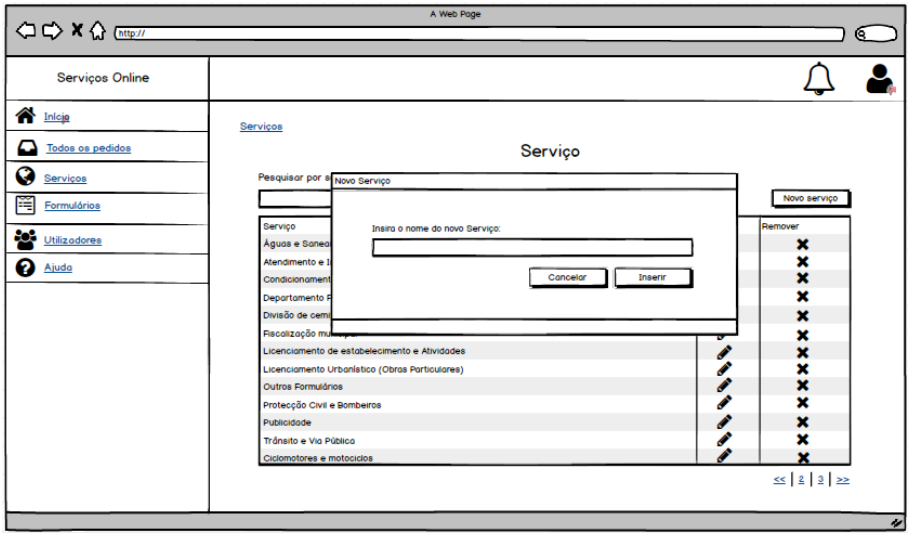


Figura 107 - Popup para adicionar novo serviço

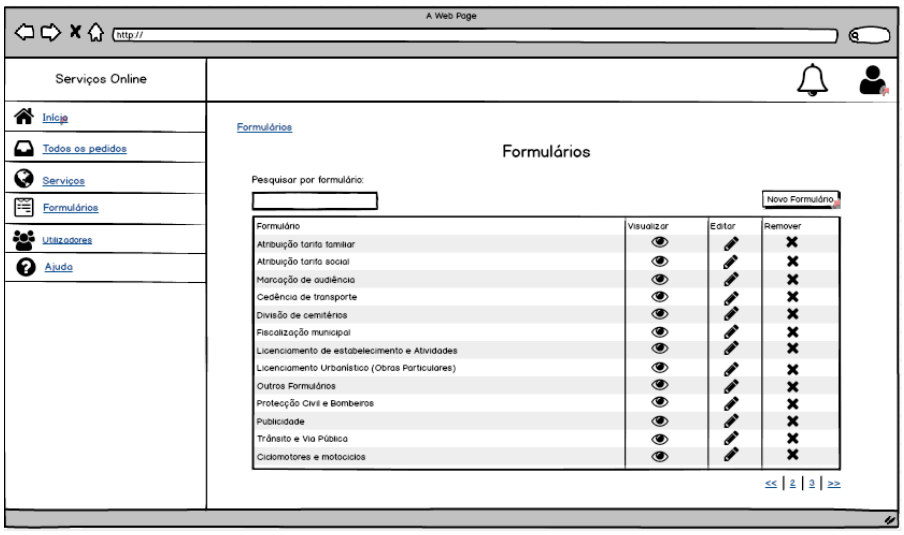


Figura 108 - Página formulários

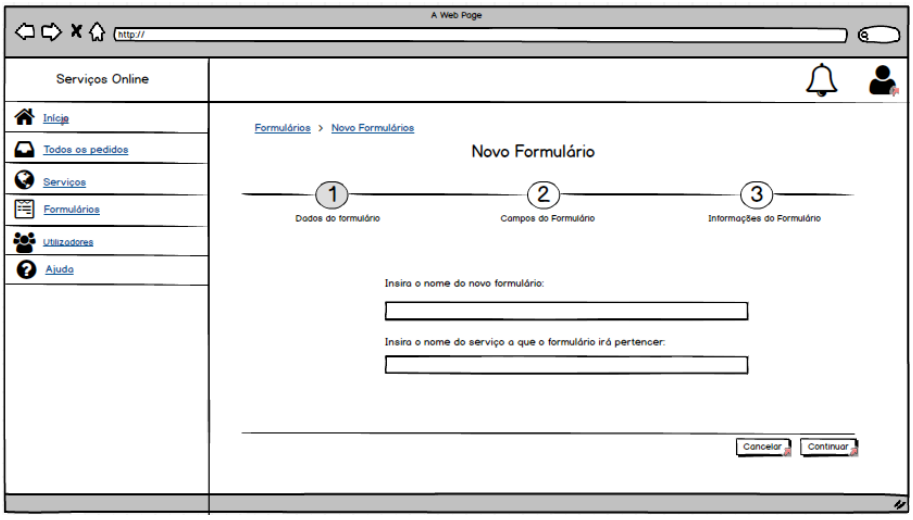


Figura 109 - Página novo formulário passo 1

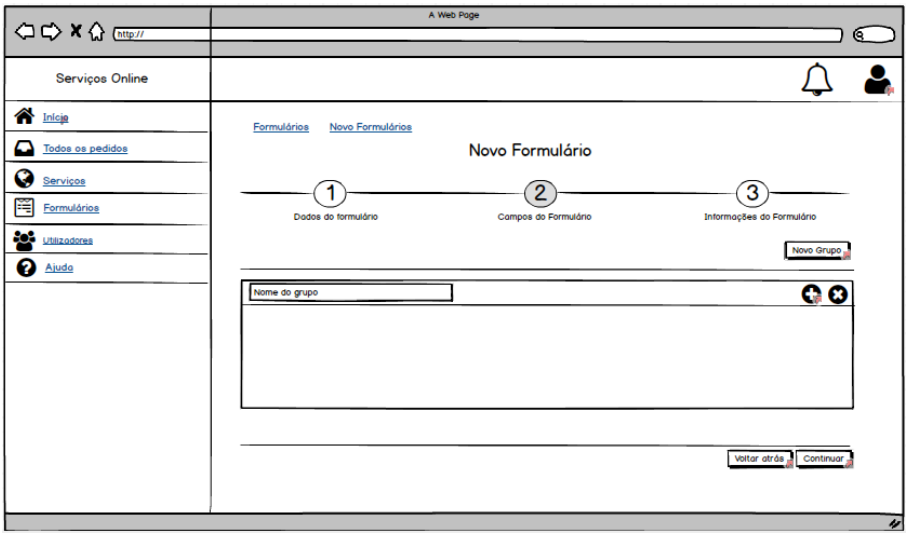


Figura 110 - Página novo formulário passo 2

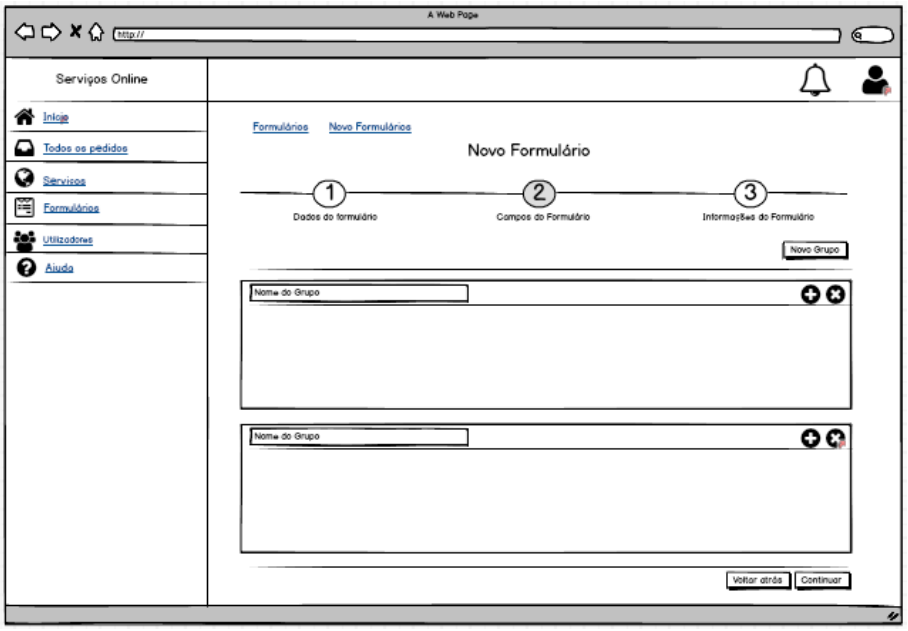


Figura 111 - Página novo formulário passo 2 (novo grupo)

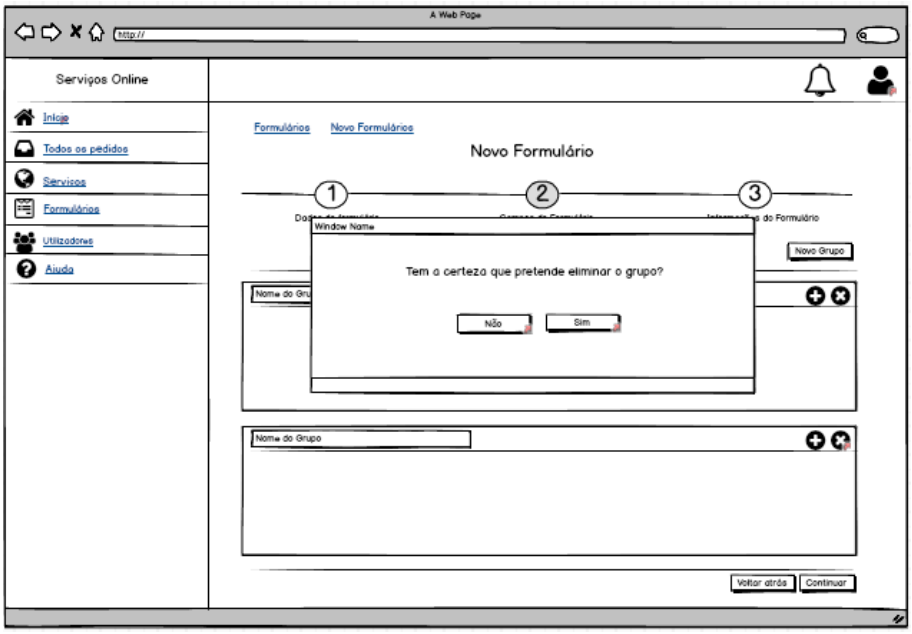


Figura 112 - Popup para eliminar grupo

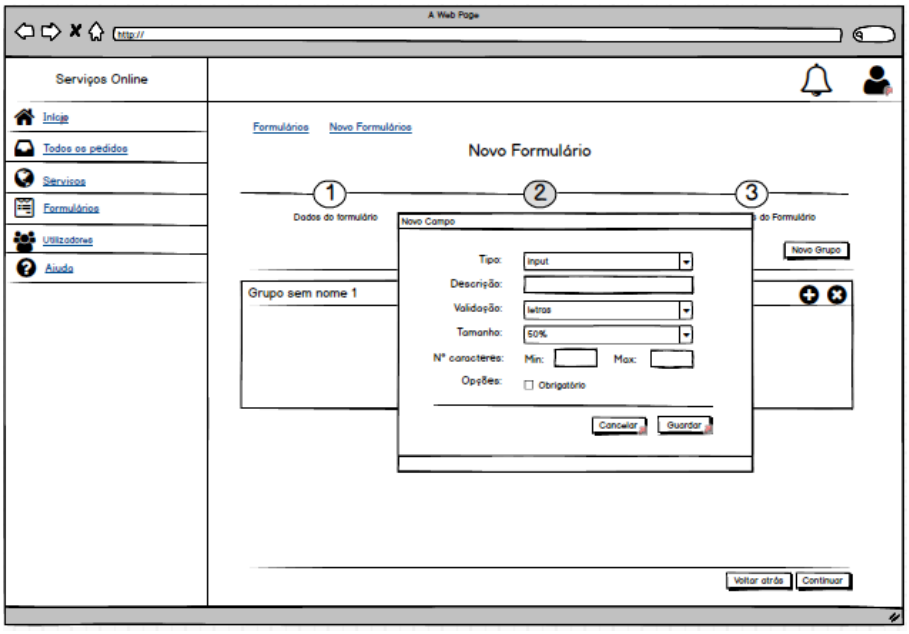


Figura 113 - Popup para adicionar campos do tipo input ao formulário

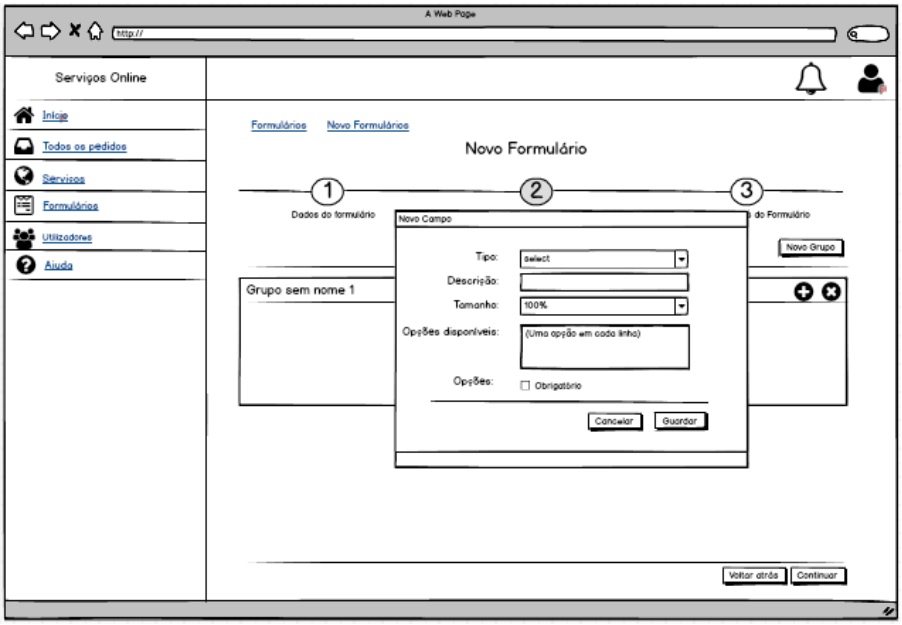


Figura 114 - Popup para adicionar campos do tipo select ao formulário

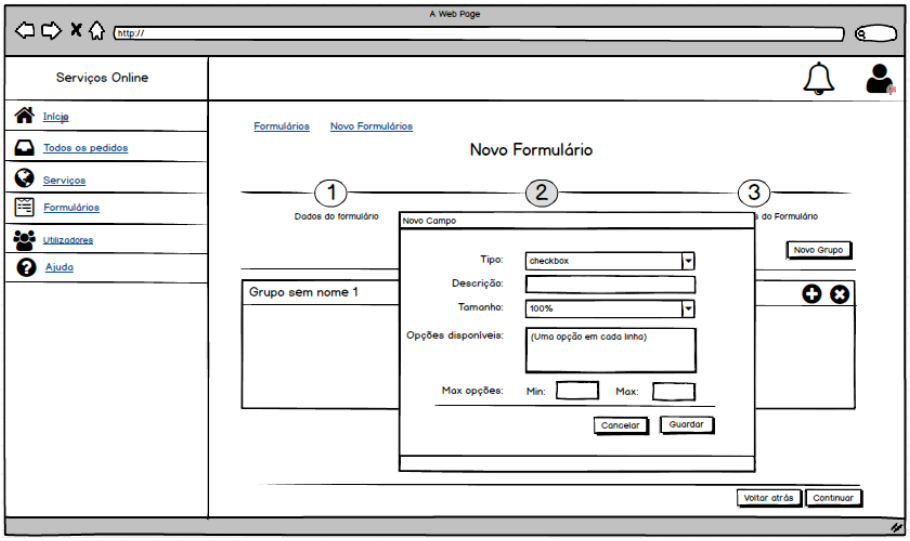


Figura 115 - Popup para adicionar campos do tipo checkbox ao formulário

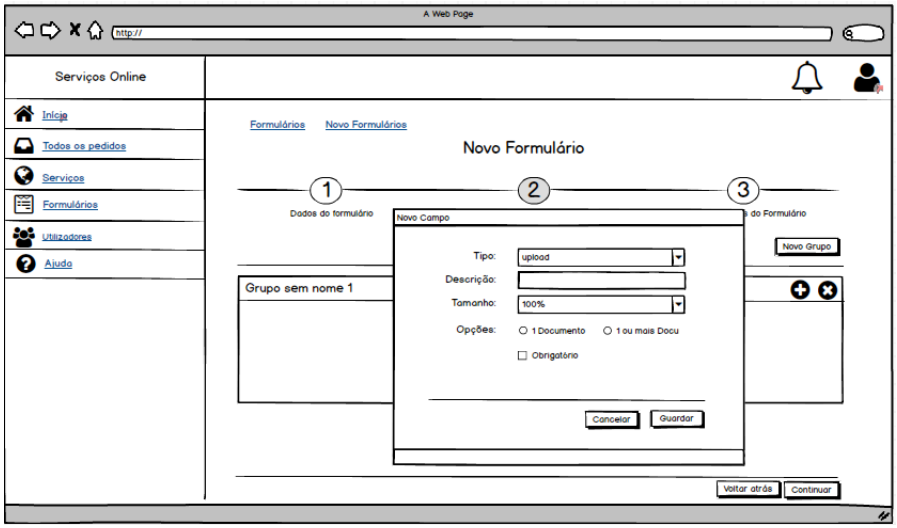


Figura 116 - Popup para adicionar campos do tipo upload ao formulário

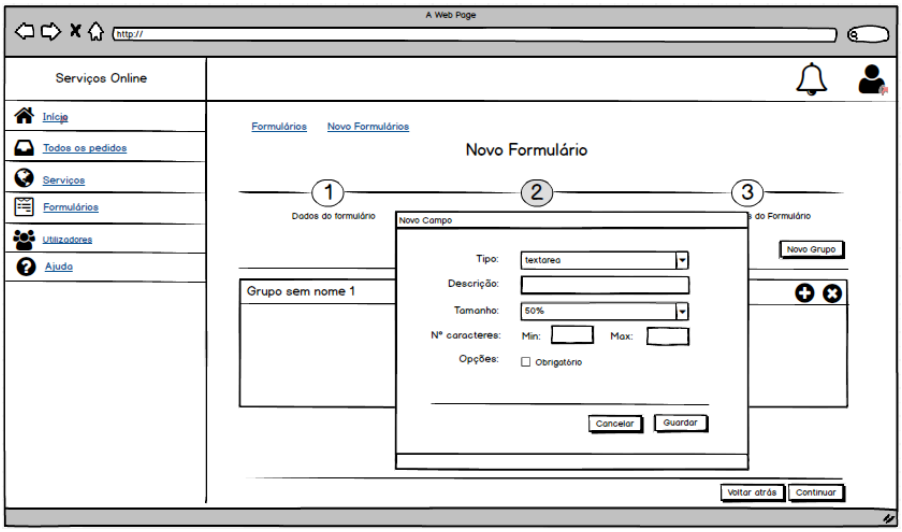


Figura 117 - Popup para adicionar campos do tipo textarea ao formulário

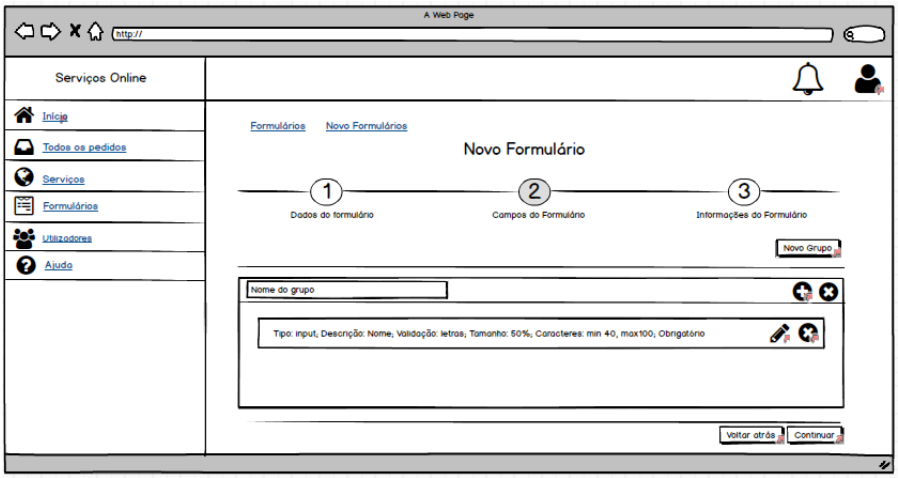


Figura 118 - Construção do formulário

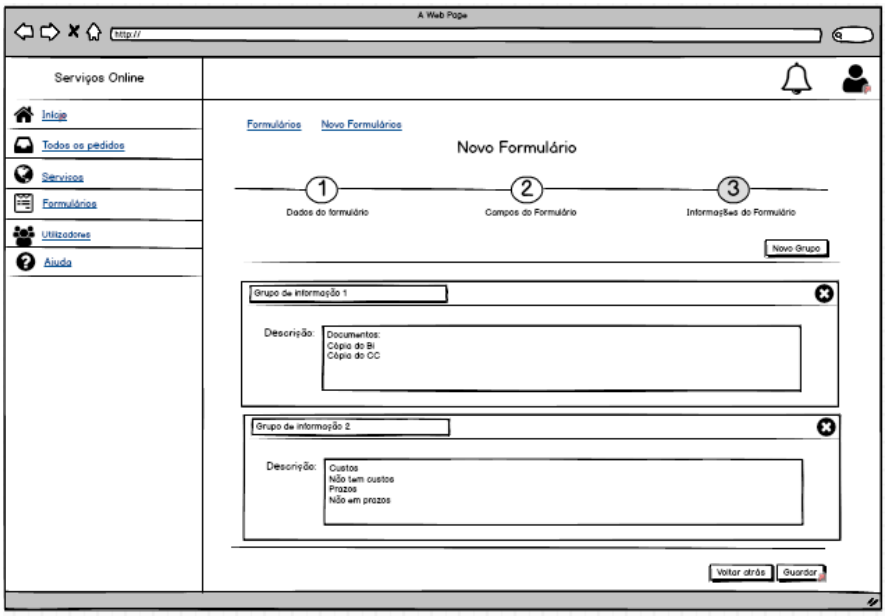


Figura 119 – Página novo formulário passo 3

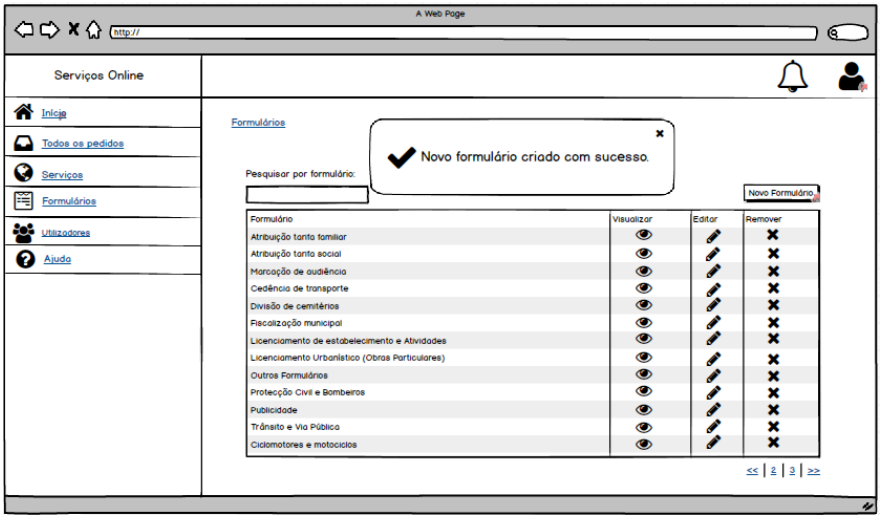


Figura 120 - Notificação formulário criado com sucesso

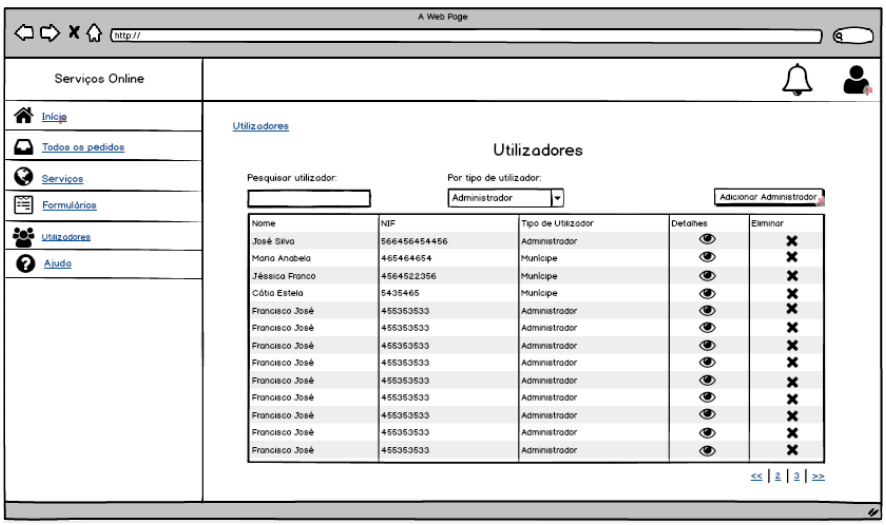


Figura 121 - Página para gerir utilizadores

8.2 Anexo B – Protótipo em Wordpress

Para o desenvolvimento deste projeto de mestrado foi utilizado como base um projeto realizado em Wordpress por antigos alunos, que era no fundo, uma continuação do trabalho que foi concretizado no âmbito da unidade curricular de Sistemas Gestores de Bases de Dados.

Como esse projeto estava a ser desenvolvido no software Wordpress, inicialmente este projeto de mestrado também seria desenvolvida nas mesmas condições para reutilizar os componentes já implementados.

Neste anexo é apresentado uma descrição geral sobre o sistema de gestão de conteúdo Wordpress e é descrito o trabalho efetuado com o mesmo.

8.2.1 Wordpress

O Wordpress é uma aplicação desenvolvida sobre PHP+MySQL em que todos os conteúdos de um *website* ficam armazenados numa base de dados, o que facilita atualizações de conteúdos bem como atualizações do próprio Wordpress.

Devido à sua capacidade de extensão através de *plug-ins*, é possível adicionar várias funcionalidades. Um dos *plug-ins* utilizados no projeto base é o Exec-PHP que permite que as páginas do Wordpress executem código PHP. Assim os componentes são desenvolvidos em páginas gravadas numa instalação do Wordpress que contêm referências a ficheiros contendo todo o código PHP e SQL necessário [46].

8.2.2 Instalação do Wordpress

Para ter acesso ao projeto, a primeira coisa a fazer foi instalar o Xampp (versão 3.2.2) e o Wordpress (versão 4.7). No decorrer da instalação foi criada uma base de dados com o nome “wordpress” para armazenar todos os dados do sistema. Após ter o Wordpress instalado, também foram instalados os *plug-ins* necessários tais como o Exec-PHP (para permitir que as páginas do Wordpress executem código PHP), *PS Disable Auto Formating* (para desativar a auto-formatação do Wordpress, necessário para se poder controlar completamente o output HTML) e o *Cabability Manager Enhanced* (para gerir as capacidades dos utilizadores).

Posteriormente, foi necessário criar ficheiros PHP com o nome de cada componente, e através da opção *Pages* que se encontra no *dashboard* do Wordpress, também foi criada uma página para cada componente, que continha a referência para os ficheiros que possuíam todo o código PHP e SQL necessário.

Por fim, foram definidas as *capabilities* para permitir o acesso aos componentes criados.

8.2.3 Base de dados

O projeto utilizado como base, tinha a particularidade de utilizar uma base de dados relacional, mas dinâmica com a capacidade de se adaptar às necessidades dos utilizadores. Desse modo, existiam no seu esquema relacional tabelas onde cada tuplo acabava por representar uma “entidade”, um “atributo” ou uma “relação”.

A base de dados relacional era constituída pelas seguintes tabelas:

- Entity type: onde cada tuplo corresponde a uma entidade;
- Relation type: onde cada tuplo corresponde a uma relação;
- Property: onde cada tuplo corresponde a um atributo;
- Property allowed value: onde cada tuplo corresponde a um valor permitido para um determinado atributo;
- Property unit type: onde cada tuplo corresponde à unidade de um determinado atributo (ex. m, cm, €);
- Entity: onde cada tuplo corresponde a uma instância de uma entidade;
- Relation: onde cada tuplo corresponde a uma instância de uma relação;
- Value: onde cada tuplo corresponde ao valor de um determinado atributo em relação a uma certa entidade;
- Custom form: onde cada tuplo corresponde à definição de um conjunto de campos que devem constar num formulário customizado;
- Custom form has prop: resultante de uma relação de muito para muitos que associa a cada formulário customizado os seus atributos.
- Um conjunto de tabelas idênticas às anteriores que servirão de *backup* e de histórico dos tuplos que forem introduzidos à medida que estes se tornam obsoletos.

O esquema relacional da base de dados está exposto na figura 122.

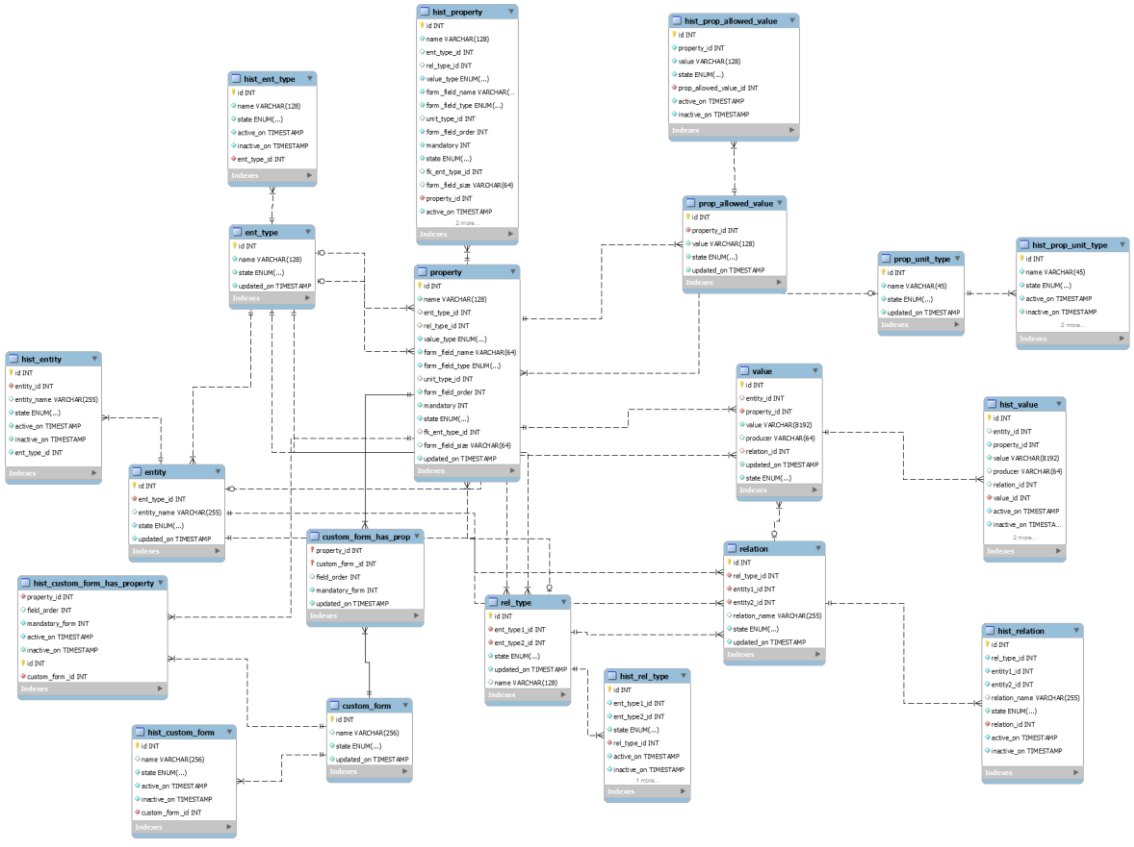


Figura 122 - Esquema relacional versão inicial

8.2.4 Adaptações realizadas

Apesar da programação do projeto utilizado como base ser orientada a objetos, foram realizadas algumas mudanças a nível de código. Nomeadamente, visto que toda a lógica estava inserida juntamente com a apresentação, começou-se por implementar o conceito MVC.

O MVC consiste na separação do código em três camadas: Modelo, Vista e Controlador [47]. Sendo assim, foram criadas essas três pastas. A pasta *model* continha os ficheiros relacionados com a lógica do programa, ou seja, métodos de manipulação de dados e consultas à base de dados. A pasta *view* continha os ficheiros relacionados com tudo o que o utilizador final visualiza, ou seja, a apresentação do sistema. E por fim, a pasta *controller* continha os ficheiros responsáveis por controlar todo o fluxo de informação do sistema.

Optou-se por implementar este conceito para facilitar a compreensão e manutenção do código, e ainda para promover a reutilização de métodos [48].

No que diz respeito a adaptações na base de dados, houve a necessidade de adicionar novas tabelas:

- Transaction type: onde cada tuplo corresponde a uma transação;
- Process type: onde cada tuplo corresponde a um processo;
- Transaction: onde cada tuplo corresponde a uma instância de uma transação;
- Process: onde cada tuplo corresponde a uma instância de um processo.

Na figura 123 encontra-se a segunda versão do esquema relacional.

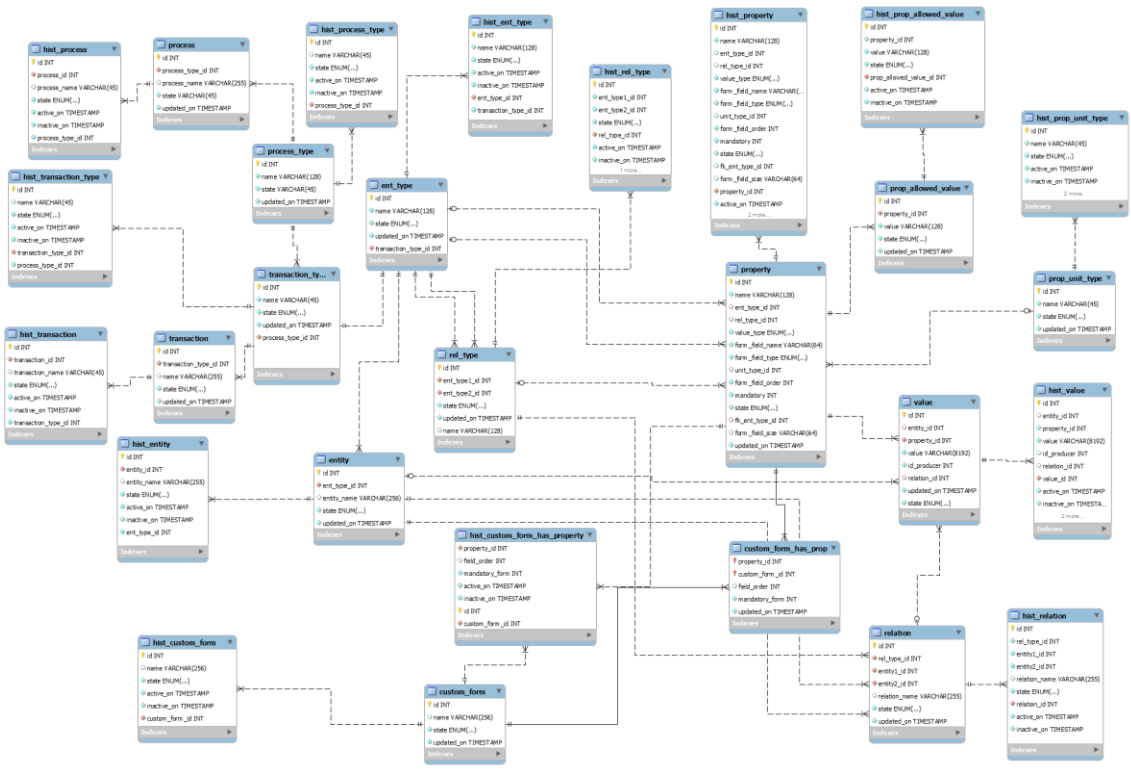


Figura 123 - Esquema relacional versão 2

Após estas adaptações procedeu-se à implementação dos componentes referentes às transações e aos processos.

8.2.5 Implementação de novos componentes

8.2.5.1 Gestão de Processos

Para começar a desenvolver este componente, primeiramente, foi criada uma página no Wordpress com o nome "Gestão de Processos".

Na página inicial deste componente é possível visualizar os processos, caso existam, e as transações que lhes estão associadas. Caso ainda não existam processos é apenas apresentado um formulário onde é possível inseri-los. Para inserir é necessário indicar o nome do processo e o seu estado. Na figura 124 é apresentada a página inicial do componente Gestão de Processos.

Figura 124 - Página inicial Gestão de Processos

Nesta página representada na figura 124 é possível alterar o estado do processo, clicando na opção Desativar ou Ativar Processo. Caso seja necessário editar um processo, apenas é possível editar o seu nome. Na figura 125 é exibido o formulário de edição referente ao componente de Gestão de Processos.

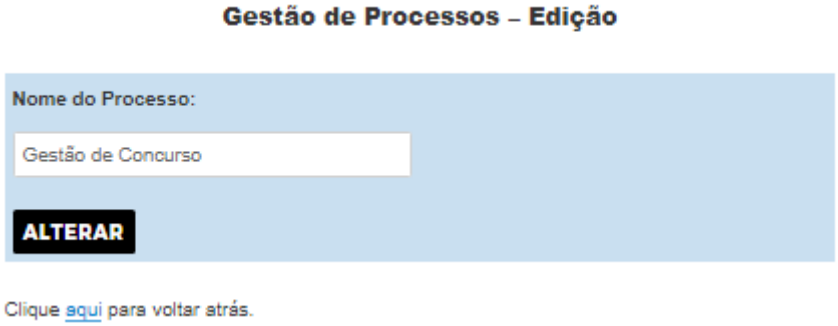


Figura 125 - Gestão de Processos (editar)

8.2.5.2 Gestão de transações

Para este componente foi criada uma página no Wordpress com o nome "Gestão de Transações".

Na página principal são apresentadas as transações existentes. Caso ainda não existam é possível inseri-las através do formulário, indicando o seu nome, estado e o processo ao qual estão associadas. A página inicial do componente Gestão de Transação está representada na figura 126.

Figura 126 – Página inicial Gestão de Transações

O estado da transação pode ser alterado clicando na opção Desativar ou Ativar Transação. No que se refere a editar transação é possível alterar o seu nome e o processo ao qual está associado. O formulário de edição referente ao componente Gestão de Transações está representado na figura 127.

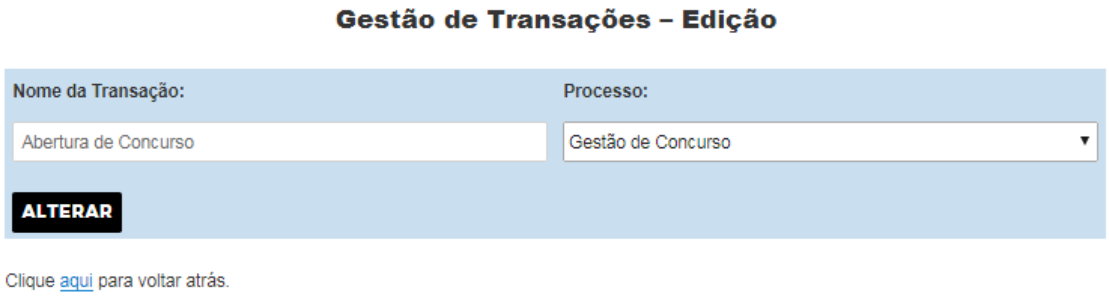


Figura 127 - Gestão de Transações (editar)

É de referir que o protótipo em Wordpress foi descontinuado quando foi decidido proceder à implementação dos componentes juntamente com outros colegas de mestrado.

8.3 Anexo C - Guia de testes de usabilidade

A finalidade deste documento é servir como guia ao participante na realização das tarefas referentes aos testes de usabilidade. O objetivo destes testes é descobrir falhas na utilização do sistema de acordo com a sua perspetiva, portanto é importante saber exatamente o que está a pensar. Por isso, peço que verbalize todas as suas dúvidas e pensamentos na realização dos testes. Estará sendo observado, no entanto apenas o sistema será alvo de avaliação. As tarefas devem ser executadas pela ordem apresentada. Agradeço a sua colaboração.

Cenário 1

O José, responsável pela entidade Clube de Atletismo XY, pretende pedir apoios à CMF para viajar com a sua equipa até a Braga onde se realiza o próximo jogo. Para isso acede à plataforma, e vê que já estão a aceitar candidaturas. Deste modo, o José faz o pedido de "Admissão de candidatura a apoios". Logo de seguida visualiza o pedido que efetuou, e repara que no painel de execução de tarefas tem tarefas para executar. Visualiza-as, prossegue para o próximo da tarefa, preenche os seus dados e finalmente conclui o pedido. De seguida o José termina sessão.

Tarefas:

- Iniciar sessão;
- Fazer pedido de "Admissão de candidatura a apoios";
- Localizar painel de tarefas de execução com tarefas para executar;
- Visualizar cada tarefa;
- Prosseguir para o próximo passo de cada tarefa;
- Preencher os dados do formulário associados a cada tarefa;
- Guardar dados;
- Terminar sessão.

Cenário 2

Ao navegar no sistema, o administrador repara que falta a propriedade "Tipo de público alvo" no tipo de entidade "Dados de apoio financeiro". Para isso, dirige-se à gestão de propriedades da entidade, e adiciona a propriedade em falta. Posteriormente, o administrador decide alterar a ordem pela qual as propriedades do tipo de entidade "Dados Gerais" serão apresentadas. Para isso, pesquisa pelo tipo de entidade e reordena as propriedades conforme a sua preferência. Finalmente, o administrador termina sessão.

Tarefas

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de propriedades";
- Clicar em "Entidade";

- Adicionar propriedade “Tipo de público alvo” no tipo de entidade “Dados apoio financeiro”, em que o estado é *active*, o tipo de valor e tipo de campo é *text*, o estado da transação é execução, o tamanho do campo é 100 e é obrigatório;
- Pesquisar pelo tipo de entidade “Dados gerais”;
- Reordenar as propriedades do tipo de entidade pesquisado anteriormente conforme preferência;
- Terminar sessão;

Cenário 3

A Ana, funcionária da CMF responsável pela seleção de candidaturas, ao entrar no sistema repara que no seu painel de tarefas de execução tem uma tarefa. Visualiza-a e segue para o próximo passo da tarefa com o objetivo de decidir se a candidatura é selecionada ou não. Ao observar os dados da candidatura, a Ana acha que é uma candidatura de interesse para a CMF, logo decide selecioná-la. Para a candidatura prosseguir para a próxima fase, é efetuado o pedido de “Decisão sobre candidatura a apoios” automaticamente pelo sistema.

Tarefas:

- Iniciar sessão;
- Visualizar a tarefa no painel de execução de tarefas;
- Prosseguir para o próximo passo da tarefa;
- Executar tarefa;
- Visualizar pedido de “Decisão sobre candidatura a apoios” realizado pelo sistema;
- Terminar sessão;

Cenário 4

O administrador do sistema pretende realizar uma pesquisa para visualizar o nome, o telefone e o email dos representantes das entidades que efetuaram o pedido de admissão de candidatura a apoios. Para isso, dirige-se à Gestão de Pesquisas, seleciona a opção “Pesquisa Dinâmica” e escolhe o tipo de entidade “Dados Gerais”. Logo de seguida seleciona as propriedades pretendidas e realiza a pesquisa. Posteriormente decide exportar os dados resultantes para formato excel e termina sessão.

Tarefas:

- Iniciar sessão;
- Localizar o menu “Gestão de Pesquisas”;
- Clicar em “Pesquisa Dinâmica”;
- Escolher o tipo de entidade “Dados Gerais”;
- Selecionar as propriedades “Nome do representante da entidade”, “Telefone do representante da entidade” e “Email do representante da entidade”;
- Efetuar a pesquisa;
- Exportar os dados para formato excel.

- Terminar sessão.

Cenário 5

O administrador do sistema tenciona efetuar uma pesquisa para visualizar todos os nomes dos representantes das entidades cujo número de sócios seja superior a 10 membros. Entretanto, recorda-se que já tinha efetuado essa pesquisa na semana passada e que a tinha guardado pois é uma pesquisa que costuma efetuar periodicamente. Portanto, para poupar tempo, dirige-se às pesquisas gravadas e efetua rapidamente a pesquisa. Por fim, termina sessão.

Tarefas:

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de Pesquisas";
- Clicar em "Pesquisas Gravadas";
- Visualizar as pesquisas gravadas;
- Efetuar a pesquisa pretendida;
- Terminar sessão.

Cenário 6

O administrador pretende efetuar uma pesquisa para visualizar todos os nomes dos representantes das entidades desportivas com mais de 10 membros que possuam a situação regularizada à administração fiscal, à segurança social e ao município do Funchal. Para isso dirige-se à Gestão de Pesquisas, seleciona a opção "Pesquisa Dinâmica", e escolhe o tipo de entidade "Dados Gerais". Logo de seguida seleciona as propriedades pretendidas, insere os respetivos valores e operadores, se necessário, e realiza a pesquisa. Como esta é uma pesquisa que lhe vai ser útil no futuro decide guardá-la. Posteriormente, o administrador termina sessão.

Tarefas:

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de Pesquisas";
- Clicar em "Pesquisa Dinâmica";
- Escolher o tipo de entidade "Dados Gerais";
- Selecionar a propriedade "Nome do representante da entidade";
- Selecionar a propriedade "Tipo de Entidade" e introduzir o valor "Desportiva";
- Selecionar a propriedade "Nº sócios filiados" e o operador maior (>) e introduzir o valor "10";
- Selecionar a propriedade "Possui a situação regularizada face à administração fiscal, à segurança social e ao município do Funchal?" e escolher o valor "Sim";
- Efetuar a pesquisa;
- Guardar a pesquisa efetuada;
- Terminar sessão.

Cenário 7

O administrador do sistema pretende efetuar uma pesquisa para visualizar a que freguesia as pessoas inseridas no sistema pertencem e qual é a sua morada específica. Para isso dirige-se à Gestão de Pesquisas, seleciona a opção "Pesquisa Dinâmica" e escolhe o tipo de entidade "Freguesia". Logo de seguida seleciona a propriedade "Nome" do tipo de entidade "Freguesia" na primeira tabela e na segunda tabela seleciona a propriedade "Nome" e a "Morada" do tipo de entidade "Pessoa". De seguida, o administrador efetua a pesquisa e decide exportar os dados resultantes para formato Excel. Por fim o administrador termina sessão.

Tarefas:

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de Pesquisas";
- Clicar em "Pesquisa Dinâmica";
- Escolher o tipo de entidade "Freguesia";
- Selecionar a propriedade "Nome" da primeira tabela;
- Selecionar a propriedade "Nome" e a propriedade "Morada" da segunda tabela;
- Efetuar pesquisa;
- Exportar resultado para formato excel;
- Terminar sessão.

Cenário 8

O administrador do sistema pretende efetuar uma pesquisa para visualizar em que data os pedidos de transporte foram efetuados pelas entidades. Para tal, dirige-se à Gestão de Pesquisas, seleciona a opção "Pesquisa Dinâmica" e escolhe o tipo de entidade "Transporte". Logo depois seleciona a propriedade "Qualidade em que faz o pedido" da primeira tabela e na tabela onde estão presentes as propriedades de relações em que a entidade Transporte está presente seleciona a propriedade "Data de pedido". Posteriormente, o administrador realiza a pesquisa visualiza os resultados e termina sessão.

Tarefas:

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de Pesquisas";
- Clicar em "Pesquisa Dinâmica";
- Escolher o tipo de entidade "Transporte";
- Selecionar a propriedade "Qualidade em que faz o pedido" da primeira tabela;
- Selecionar a propriedade "Data de pedido" da tabela que contém as propriedades de relações em que a entidade Transporte está presente;
- Efetuar pesquisa;
- Terminar sessão.

Cenário 9

O administrador do sistema tenciona efetuar uma pesquisa para visualizar todas as informações das entidades que efetuaram pedidos de transporte e ainda deseja saber com que intenção foram feitos esses pedidos, e quem são as pessoas responsáveis pelo transporte e respetivo contacto telefónico. Para tal, dirige-se à Gestão de Pesquisas, seleciona a opção "Pesquisa Dinâmica" e escolhe o tipo de entidade "Entidade". Seguidamente seleciona todas as propriedades da primeira tabela e na última tabela seleciona as propriedades "Qualidade em que faz o pedido", "Pessoa responsável" e "Contacto pessoa responsável". Posteriormente efetua a pesquisa, visualiza os resultados e termina sessão.

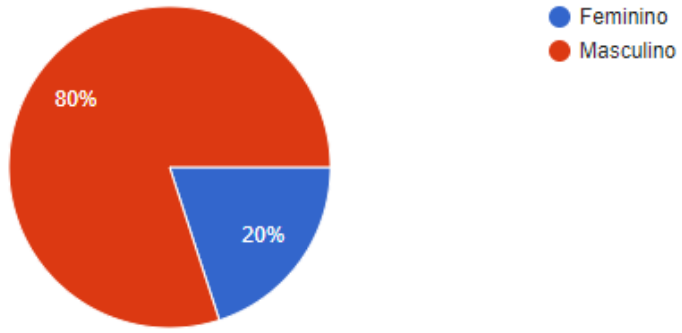
Tarefas:

- Iniciar sessão;
- Dirigir-se ao menu "Gestão de Pesquisas";
- Clicar em "Pesquisa Dinâmica";
- Escolher o tipo de entidade "Entidade";
- Selecionar todas as propriedades da primeira tabela;
- Selecionar as propriedades "Qualidade em que faz o pedido", "Pessoa responsável" e "Contacto pessoa responsável" da última tabela;
- Efetuar pesquisa;
- Terminar sessão.

8.4 Anexo D – Resultados do Questionário

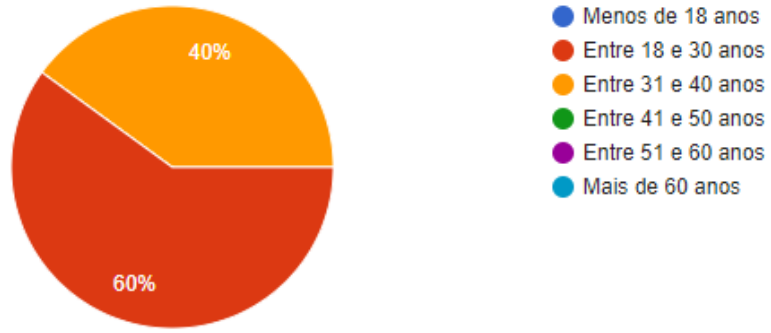
Escolha o seu género

5 respostas



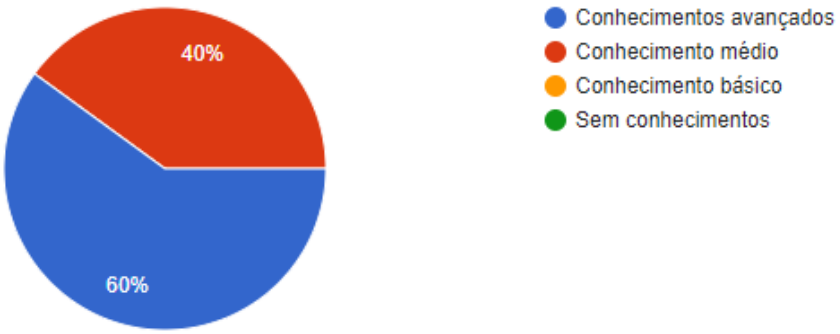
Indique a sua faixa etária

5 respostas



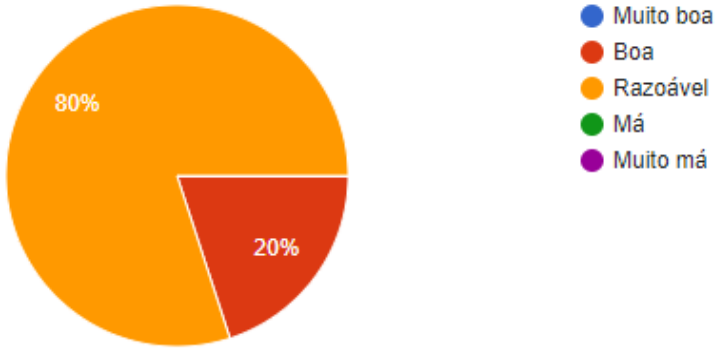
Como classifica os seus conhecimentos na ótica do utilizador?

5 respostas



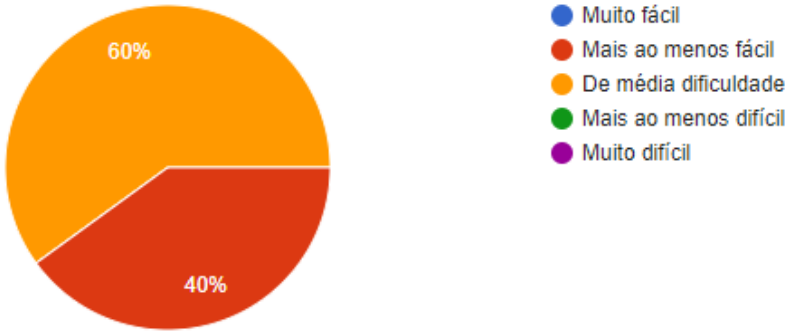
Como avalia a aparência do sistema?

5 respostas



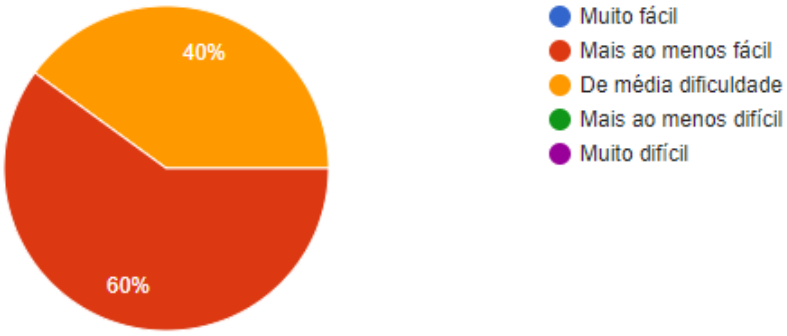
Achou o sistema fácil de utilizar?

5 respostas



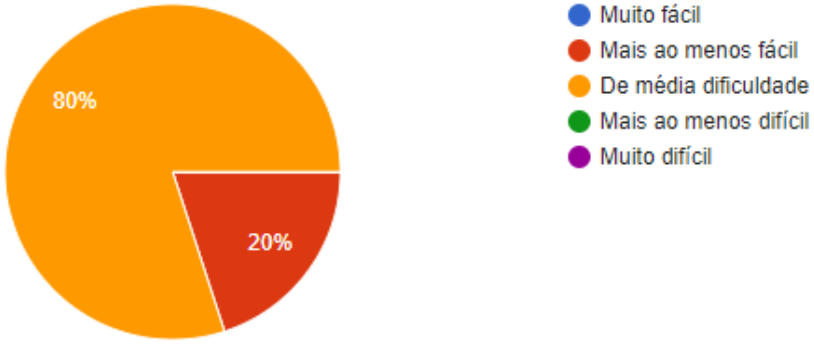
É fácil de aprender a utilizar o sistema?

5 respostas



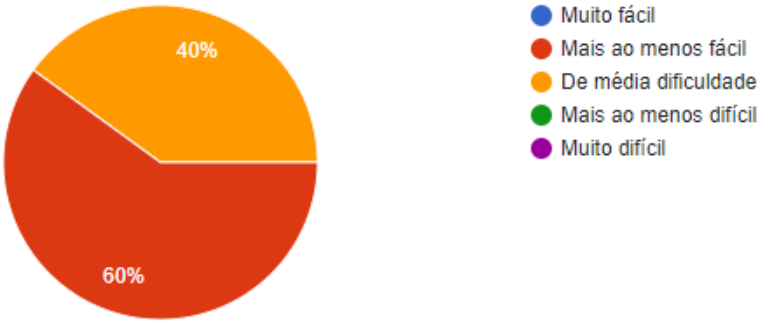
Foi fácil encontrar a informação desejada?

5 respostas



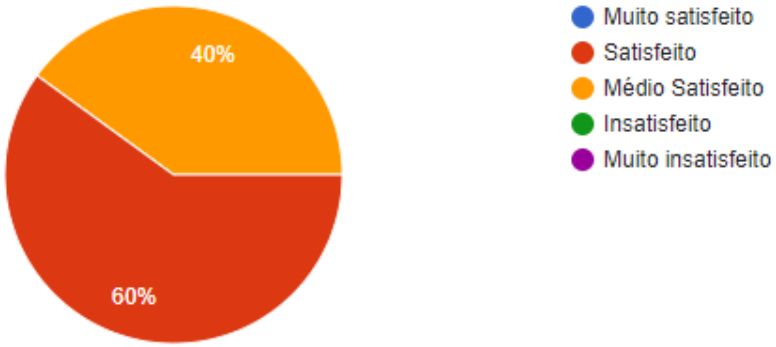
Foi fácil atingir o objetivo utilizando o sistema?

5 respostas



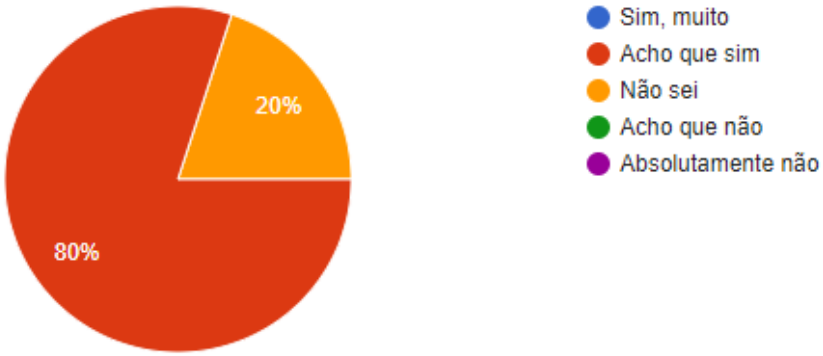
Está satisfeito com a facilidade de realização dos cenários?

5 respostas



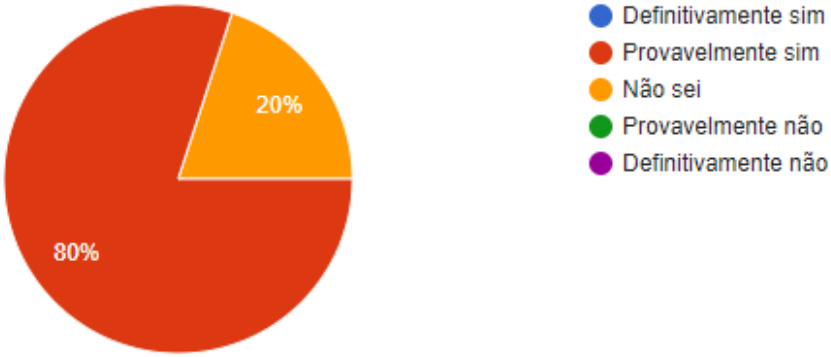
Acha o sistema útil?

5 respostas



Recomendaria o sistema a outras pessoas?

5 respostas



Observações ou melhorias

2 respostas

A parte visual e disponibilização de algumas funcionalidades devem ser trabalhadas de forma a tornar o sistema mais intuitivo.

Para prótotipo, parece-me bem!