Universidade de Aveiro

2015

Departamento de Eletrónica, Telecomunicações e Informática

Tiago Manuel Ferreira Gandarez

Plataforma para apoio à formação do Sistema de Informação da Justiça de Cabo Verde

Universidade de Aveiro
2015

Departamento de Eletrónica, Telecomunicações e Informática

Tiago Manuel Ferreira Gandarez

Plataforma para apoio à formação do Sistema de Informação da Justiça de Cabo Verde

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistemas de Informação, realizada sob a orientação científica do Doutor Cláudio Teixeira, Equiparado a Investigador Auxiliar e do Doutor Joaquim de Sousa Pinto, Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

Dedico este trabalho aos meus pais, aos meus irmãos e à Stéphanie por todo o apoio e esforço despendido.

o júri

presidente

Prof. Doutor José Luís Guimarães Oliveira
Professor Associado da Universidade de Aveiro

vogais

Prof. Doutor André Frederico Guilhoto Monteiro
Professor Auxiliar Convidado no Instituto Superior Miguel Torga

Doutor Cláudio Jorge Vieira Teixeira
Equiparado a Investigador Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro

agradecimentos

Quero agradecer em primeiro lugar à minha mãe e ao meu pai, pois reconheço os enormes sacríficos feitos por eles, para que eu conseguisse aproveitar todas as oportunidades da melhor forma. Quero agradecer também aos meus dois irmãos, por todas as ajudas que me deram desde o dia em que nasci até ao dia de hoje. Agradeço também à Stéphanie, pelo apoio e motivação dada neste últimos anos.

Ao professor Cláudio Teixeira e ao professor Joaquim Sousa Pinto, não só pela oportunidade que me deram de entrar no seu grupo de trabalho, mas também por me ajudarem a evoluir todos os dias.

Aos meus colegas do IEETA, não esquecendo os que já saíram, pelo paciência e pela forma como me fizeram evoluir.

Aos amigos que fui fazendo no meu percurso académico e aos amigos de longa data por todas as boas intervenções que ajudaram no meu percurso.

E gostava de também deixar uma palavra de agradecimento ao clube Lobitos Futsal e a todos os seus (atuais e antigos) elementos, pelos princípios que me deram e pela forma que me ensinaram a encarar os desafios.

O meu muito obrigado a todos.

palavras-chave Formação, SIJ, Cabo Verde, Justiça, Sistema de Informação, Software

resumo Esta dissertação surgiu com a necessidade de formar os utilizadores do Sistema de Informação da Justiça de Cabo Verde. O sistema é complexo e composto por utilizadores com diferentes funções, pelo que existe alguma dificuldade na formação dos utilizadores para o desempenho das suas tarefas. Desta forma, foi necessário desenhar e implementar um sistema de informação que consiga formar os utilizadores de acordo com as suas responsabilidades no Sistema de Informação da Justiça, permitindo criar, configurar e acompanhar as formações em questão. Pretende-se ainda que este sistema possa ser utilizado em ambiente de sala de aula, bem como num ambiente particular.

keywords Training, SIJ, Cape Verde, Justice, Information System, Software

abstract This dissertation came up from the need to train users of the Cape Verde Justice Information System. The system is complex and composed by users with different roles, so there is some difficulty in training users to perform their tasks. Thus, it was necessary to design and implement an information system that can train users according to their responsibilities in the Information System of Justice, allowing the creation, configuration and monitorization of the training in question. It is also intended that this system can be used in a classroom environment as well as in a particular.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

API	Application Programming Interface
CSS	Cascading Style Sheets
DTD	Document Type Definition
FAQ	Frequently Asked Questions
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
MVC	Model-View-Controller
REST	Representational State Transfer
SaaS	Software as a Service
SGBD	Sistema de Gestão de Base de Dados
SGML	Standard Generalized Markup Language
SIJ	Sistema de Informação da Justiça
SOA	Service-Oriented Architecture
SQL	Structured Query Language
TFS	Team Foundation Server
XML	eXtensible Markup Language
XSD	XML Schema Definition
W3C	World Wide Web Consortium
WCF	Windows Communication Foundation
WSDL	Web Service Definition Language

1 Introdução

1.1 Motivação

Nos dias de hoje encontramos sistemas cada vez mais complexos. Sistemas que grandes companhias utilizam para as ajudar a realizar as suas tarefas de gestão e de produção. Por vezes encontram-se sistemas tão grandes e complexos que não existe ninguém na empresa que os domine por completo. São estes problemas que trazem a necessidade de quem concebe e desenvolve os sistemas de tornar a sua utilização o mais simples possível. No entanto, por vezes até o mais simples ainda é complexo. E surge aí a necessidade de alguém garantir que os utilizadores do sistema conseguem ultrapassar as barreiras da aprendizagem, conseguindo assim aproveitar realmente as suas capacidades, tirando dessa forma o proveito que o sistema se propõe a dar. À medida que o tempo passa surgem novas tecnologias e metodologias mais inovadoras que proporcionam uma melhor e maior produção para uma organização. No entanto, alguns problemas podem estar relacionados com a implementação de inovação, como por exemplo a dificuldade de adaptação a novas funcionalidades por parte dos funcionários. Isto pode traduzir-se em tarefas executadas de forma deficiente, e consequente desmotivação por parte do funcionário. Assim, a implementação de novas funcionalidades, se for mal aplicada pode tornar-se em custos sem retorno para a empresa.

Desta forma, uma solução viável é a realização de formações aos funcionários das organizações de modo a que estes consigam extrair o máximo proveito dos sistemas que utilizam para o desempenho das suas tarefas, e assim fazer com que a organização consiga ver o seu investimento retornado.

1.2 Sistema da Informação da Justiça

O Sistema de Informação da Justiça (SIJ) começou a ser desenvolvido em 2009 pela Universidade de Aveiro. Desde aí, a equipa tem vindo a crescer, contando atualmente com cerca de 11 colaborares dedicados às correções de eventuais problemas e ao também ao desenvolvimento de novas funcionalidades.

O SIJ é um sistema Web usado nos tribunais de Cabo Verde, desenvolvido para dar suporte ao fluxo dos processos de um tribunal permitindo assim ajudar na gestão e desmaterialização de processos

[1]. Assim sendo, uma vez informatizados todos os dados dos processos, torna-se mais fácil e eficiente a gestão dos mesmos, trazendo uma melhoria à justiça em Cabo Verde.

Relativamente aos utilizadores do sistema, todos têm uma função associada e só dessa forma é possível alguém autenticar-se no sistema. A cada função estão associadas diferentes permissões. Assim, cada utilizador tem as suas tarefas tendo que as realizar para que o processo esteja em andamento e para que outros utilizadores dependentes desse processo possam trabalhar. Algumas das funções existentes no sistema são as funções de juiz, de procurador, de advogado e de funcionários administrativos [1]. Na **Figura 1.1** podemos ver três utilizadores com funções diferentes, com exemplos de tarefas que cada um pode fazer tendo em consideração a sua função.

O sistema é composto por uma aplicação Web e por um serviço de gestão de *workflows,* sendo então um sistema composto por duas aplicações distintas que comunicam entre si com base numa arquitectura orientada a serviço (SOA). Estas duas aplicações estão desenvolvidas sobre a mesma lógica de negócio e sobre a mesma base de dados. A aplicação Web tem como objectivo fornecer uma interface gráfica ao utilizador final, de modo a tornar a sua interação com o sistema o mais amigável possivel. Assim, todas as interações que um utilizador tem com o SIJ são através desta aplicação Web. Relativamente ao serviço de gestão de *workflows,* este foi desenhado com o intuito de gerir todas questões relativas ao andamento dos processos. É através deste serviço, tendo em consideração o tipo de processo, que se consegue fazer uma gestão das fases e dos estados que o compõem. É também este serviço que gere as permissões que são dadas aos utilizadores para um determinado processo numa determinada fase [1] [2].

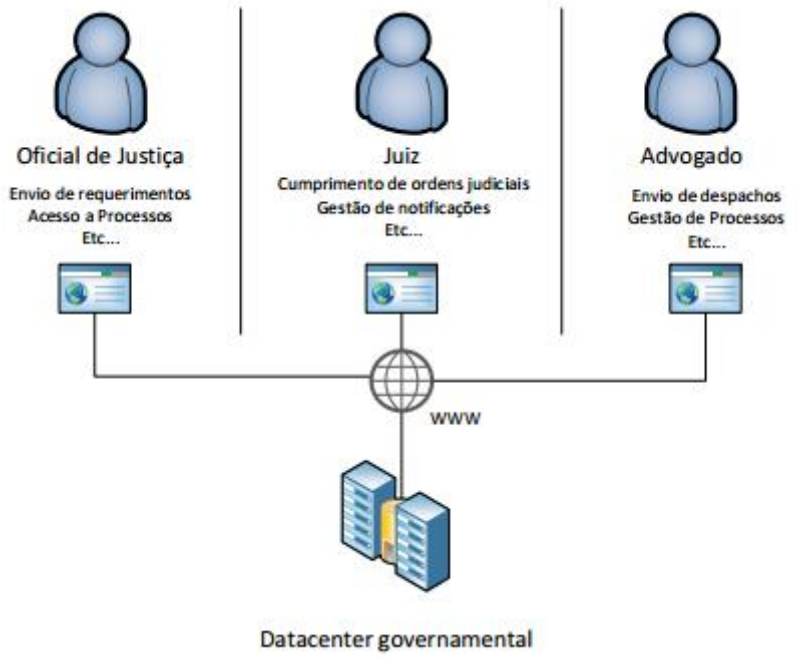

Figura 1.1 - Roles SIJ

1.3 Objetivos

Nos dias de hoje, deparamo-nos com sistemas de informação bastante grandes e complexos. Assim, a ideia da formação junto dos utilizadores tem de ser cada vez mais tida em consideração, pois pode ser isso que vai definir o sucesso de um determinado produto. Se existe capacidade para construir sistemas tão poderosos também deverá existir capacidade de criar um sistema paralelo que auxilie a aprender e perceber como funcionar o sistema principal.

Contextualizando-nos, o Sistema de Informação da Justiça pode ser considerado um desses sistemas bastante complexos, ainda mais quando utilizado num país como Cabo Verde onde a literacia informática é baixa quando comparada com a literacia informática no mundo ocidental. Foi aqui que surgiu a oportunidade de criar um sistema de formação que conseguisse atenuar ou mesmo eliminar a dificuldade que possa existir para os utilizadores do SIJ.

Assim, os objetivos do sistema da formação passam por conseguir proporcionar a quem desenvolveu o SIJ, a capacidade fazer chegar aos utilizadores finais uma forma destes conseguirem aprender a realizar as suas tarefas no SIJ. Esse sistema deve conseguir responder às necessidades que sejam colocadas na realização de formações a grupos de pessoas, bem como às necessidades que sejam colocadas na realização de tarefas por parte dos formandos por iniciativa própria. Assim, o sistema da formação deve permitir a criação prévia de *templates* de tarefas para que na hora da

formação estejam prontos para simplesmente iniciar de modo a que os formandos possam fazer as tarefas programadas.

Sendo o SIJ um sistema composto por vários tipos de utilizadores e tarefas associadas a cada um destes tipos, essas tarefas são muitas vezes dependentes de tarefas realizadas por outros utilizadores. Isto faz com que certos utilizadores só possam realizar as suas tarefas depois dos outros utilizadores fazerem as que lhes competem, o que implica que estes não podem fazer nada até que os outros concluam as suas tarefas. Num contexto de formação, não seria uma boa prática dar tarefas a formandos que estivessem dependentes de terceiros, pois iria provocar uma má gestão de tempo. Desta forma, uma das ideias deste sistema é permitir a criação de formações para utilizadores com uma certa função. Então, como existem tarefas que dependem de outras, o sistema deve conseguir fazer uma simulação dessas tarefas, de modo a que as tarefas para os utilizadores que estão a ser formando estejam prontas a ser realizadas instantaneamente aquando o começo da formação. Outro aspeto importante é o facto do sistema da formação ser também responsável por fazer uma monitorização das tarefas que vão sendo realizadas pelos formandos, avaliando os dados inseridos por estes, dando a conhecer tanto ao formando como aos administradores os resultados das mesmas. Assim, os formandos podem realizar as suas tarefas sem ser num ambiente de formação, como por exemplo em casa, conseguindo saber se estão a ser bem ou mal executadas.

Assim, o desenvolvimento do sistema da formação foi dividido em duas partes. Uma parte responsável por simular as tarefas que não devem ser realizadas pelos formandos e por fazer também um acompanhamento das tarefas que os formandos vão realizando. E outra parte, responsável por permitir a criação e gestão de *templates* de tarefas para formações, bem como a criação e gestão de configurações de quem deve fazer determinadas tarefas. Deve ainda permitir que os administradores e os formandos possam ver a partir de um sistema Web o estado de cada tarefa realizada e o estado global do formando numa determinada formação. Foi esta segunda parte que proporcionou a realização desta dissertação.

1.4 Contribuição

Como foi dito no ponto anterior, o foco desta dissertação passa pelo desenvolvimento da parte relativa à gestão de *templates*, que são constituídos por um conjunto de tarefas ordenadas para serem realizadas pelos formandos ou pelo sistema de simulação. Este sistema deve permitir que as tarefas desses *templates* possam ser configuradas no que diz respeito aos *roles* que as devem

realizar. Deve ser ainda permitido por parte deste sistema o arranque de formações, podendo acompanhar o estado de cada tarefa à medida que estas são realizadas.

Assim, o sistema foi desenvolvido com objetivo de ser o mais intuitivo e funcional possível, trazendo desta forma uma mais-valia para as pessoas que estão responsáveis pela gestão das formações, permitindo que estas consigam criá-las com *roadmaps* bem definidos. Este sistema visa oferecer ainda a possibilidade dos gestores da formação poderem configurar um determinado *template* consoante as necessidades da formação. Isto é, se a formação tem o objetivo de formar juízes, as tarefas que não são da competência destes podem ser configuradas para serem simuladas pelo sistema, evitando assim que os formandos juízes façam tarefas que não lhes competem ou que tenham de esperar que alguém as faça. É importante salientar o facto de poderem serem feitas varias configurações de perfis para um determinado *template*. O sistema deve ainda permitir que tanto os administradores como os formandos possam iniciar formações com base em perfis já configurados, bem como acompanhar o estado das tarefas dessas formações.

1.5 Organização do Documento

Este documento encontra-se dividido em cinco capítulos.

Neste primeiro capítulo, temos o enquadramento deste trabalho. No segundo capítulo está descrito algum conhecimento teórico relativamente à formação. Ainda no segundo capítulo são abordados três sistemas que atuam nessa área.

Relativamente ao terceiro capítulo são abordados os requisitos, a arquitetura e o modelo de dados do projeto. Já no quarto capítulo é feita uma descrição dos casos mais importantes na implementação do sistema e ainda é descrito o sistema com recurso a imagens tiradas deste.

No capítulo cinco são apresentadas as conclusões retiradas bem como o trabalho que pode ser feito no futuro relativamente a este sistema.

2 Estado de Arte

Neste capítulo serão abordados os tipos de formações possíveis de implementar numa organização, bem como as suas vantagens, formas de organização e ainda as consequências que a falta de formação pode trazer. É também abordado o ponto de vista do funcionário relativamente à sua formação. Por último são apresentadas três tecnologias que foram desenvolvidas com o objetivo de ajudar os utilizadores quando chegam pela primeira vez a uma determinada plataforma Web, não tendo assim o conhecimento de como a utilizar.

2.1 Características da Formação

2.1.1 Formação nas Organizações

Nos dias de hoje a formação é bastante importante para uma empresa. A sua existência nas empresas é decisiva para a qualidade dos seus produtos ou serviços bem como para o seu volume de negócios.

No que diz respeito a implementar formação numa organização é necessário ter em conta todo o processo. Na **Figura 2.1** temos um diagrama composto por cinco passos que são importantes para as organizações que querem implementar formação.

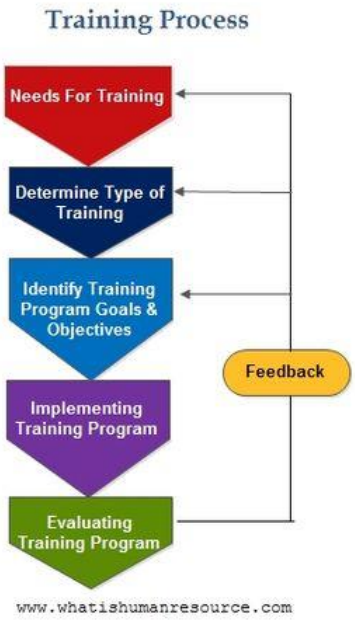

Figura 2.1 - Processo formação **[3]**

O primeiro passo tem como objetivo verificar a necessidade de formação para os funcionários da organização em questão. Ou seja, se existe algum aspeto que a formação venha melhorar. O segundo passo visa determinar qual o tipo de formação que a organização necessita. Isto é, que tipo de formação vai resolver os problemas ou vai trazer uma melhoria. Depois de identificadas as necessidades deve-se proceder à identificação das metas e dos objetivos de modo a que a formação esteja o mais direcionada possível para os requisitos definidos. O quarto passo é relativo à implementação da formação com base nos requisitos anteriormente definidos. Assim, é necessário nomear um instrutor com grande conhecimento do tema da formação, para poder passar da melhor forma o conhecimento para os formandos. Por último deve ser feita uma avaliação da formação de modo a se saber se os requisitos definidos nos passos anteriores foram cumpridos. Assim, é possível melhorar continuamente este processo em futuras formações [3].

São conhecidos dois tipos de formação: *on-the-job training* e *off-the-job-training*. Na **Tabela 1** estão apresentadas diferenças entre os dois tipos [4].

Diferenças entre *on-the-job training* e *off-the-job training*	
on-the-job training	***off-the-job training***
No ambiente real do trabalho. Aprender fazendo.	Fora do ambiente de trabalho.
O treino não impede a produção.	Produção interrompida durante a formação.
Tem o objetivo de ensinar as melhores práticas para uma tarefa específica.	Tem o objetivo de ensinar conceitos básicos e mais gerais.
Normalmente quem dá a formação são os funcionários experientes.	Normalmente quem dá a formação são formadores contratados.
Mais simples e mais barato.	Mais caro.
Rotação de atividade e aprendizagem é o método mais utilizado.	Aulas e exercícios de simulação é o método mais utilizado.
Normalmente para tarefas relacionadas com a produção.	Normalmente para tarefas relacionadas com gestão e outros tipos exceto tarefas de produção.

Tabela 1 - on-the-job training VS off-the-job training

Com base na **Tabela 1** podemos dizer que quando um funcionário aprende a fazer as suas tarefas no próprio local de trabalho estamos perante o tipo *on-the-job training.* Este método, muitas vezes utilizado tem a grande vantagem de dar o conhecimento em condições reais de trabalho, fazendo com que o funcionário faça já as tarefas para as quais é pago.

Podemos ver abaixo cinco métodos associados ao tipo *on-the-job training* [5] [6] [7]:

- **Rotatividade**: O funcionário recebe formação em vários tipos de tarefas. Este método dá a oportunidade ao funcionário de ficar a conhecer as tarefas de outros funcionários, percebendo assim os eventuais problemas que eles possam vir a ter. Este método fortalece o relacionamento entre funcionários.
- **Treino**: O formando é colocado junto de um funcionário mais experiente de modo a que este o possa supervisionar. A vantagem deste método é a rápida identificação de dificuldades do formando por parte do funcionário mais experiente. No entanto também existe a desvantagem do formando aprender más práticas comuns com esse funcionário.
- ***Mentoring***: Este método é focado na atitude do formando e ajuda a descobrir fraquezas deste, de modo a que haja um maior foco nelas.
- **Instrução técnica do trabalho**: É neste método que o instrutor explica passo-a-passo como as tarefas devem ser feitas. Em primeiro lugar prepara o formando com uma vista geral de como as tarefas devem ser feitas e quais os seus objetivos finais. A seguir, apresenta uma tarefa ao formando, deixando que este a realize de forma autónoma, para que no final possa ser dado *feedback* acerca da maneira como o formando realizou a tarefa.
- **Aprendizagem continua**: Este método é utilizado em áreas onde o funcionário deve ter grandes habilidades para fazer as suas tarefas. Assim, os formandos trabalham juntamente com os funcionários mais experientes por um longo período de tempo até adquirirem a habilidade necessária.

Abordando as vantagens que o tipo *on-the-job training* traz, podemos enumerar algumas, como o facto de a formação ser dada no contexto real de trabalho. Assim a aprendizagem do formando torna-se mais eficiente devido a este aprender com a experiência, fazendo também que o formando fique mais motivado. Este método revela-se também mais barato. No entanto também existem desvantagens, como o facto de ser um sistema não tão bem organizado. Além disso os formadores, que são muitas vezes funcionários mais experientes, podem não ter a experiência ou aptidão pretendida para dar formação. Importa ainda referir que quando uma formação é mal organizada, esta pode trazer riscos de segurança [5].

Relativamente ao tipo *off-the-job training* o local da formação é fora do local de trabalho, num local designado para o efeito. Neste tipo de formação os formandos estão afastados das distrações que podem existir no local de trabalho. No entanto a aquisição do conhecimento para aplicar nas tarefas pode não ser tão eficiente. Este tipo de formação também tem associados alguns métodos que podemos ver abaixo [7] [8]:

- **Aulas e conferências**: Este método é baseado em aulas onde existe um orador que explica as tarefas. Isto é, como elas devem ser feitas e qual o objetivo de cada uma. Este método tem a vantagem de poder ser dado a grupos de pessoas o que vai diminuir os custos.
- **Formação *Vestibule:*** Este método é mais adequado para funcionários de escritório e funcionários que trabalham com certas ferramentas. Então este método procura trazer os equipamentos de trabalho para o local (local diferente do local de trabalho) onde a formação vai decorrer. Assim, os formandos apresentam-se menos nervosos relativamente à sua prestação, podendo fazer as suas tarefas sem sentirem pressão e sem medo de errar. Além disso, como o ambiente é parecido com o ambiente real, o formando consegue reter o conhecimento facilmente.
- **Simulação**: Este método tenciona simular na totalidade o ambiente real de modo a fazer com que o formando sinta que está a fazer as suas tarefas no mundo real. Este método é muitas vezes usado para dar formação a pessoas que tem tarefas com máquinas ou equipamentos muito caros ou mesmo tarefas perigosas.
- **Casos de estudo**: Este método tem o objetivo de dar um documento de texto aos formandos com a descrição de uma situação que aconteceu no passado, dentro ou fora da empresa. A seguir os formandos devem ler, analisar e tirar as suas próprias conclusões. No final o caso é discutido pelo formador, onde este apresenta as suas conclusões.

As vantagens associadas a este tipo de formação são o facto de os formandos conseguirem obter conhecimento suficiente para realizarem as suas tarefas. É um sistema organizado sendo que programas criados corretamente podem trazer grande valor. Relativamente às desvantagens, este tipo não consegue simular tão bem o ambiente real de trabalho relativamente ao outro. É portanto um ambiente menos natural. Além disso torna-se mais caro e os formandos podem-se revelar menos motivados [8].

Relativamente à escolha do tipo de formação, deve-se ter em conta fatores como o objetivo principal da formação, o orçamento disponível, o número de formandos, o tempo disponível, entre outros.

2.1.2 Formação em Contexto de Trabalho (on-the-job training)

A formação em contexto de trabalho pode ser realizada de duas maneiras diferentes: a formação planeada e a não planeada. A formação planeada acontece no posto de trabalho e consiste em atividades organizadas desenvolvidas durante o trabalho do funcionário. Aqui existe a presença de uma pessoa que demonstra como se faz uma determinada tarefa e depois, no caso de ser necessário faz uma orientação [9].

A formação não planeada acontece também no posto de trabalho do funcionário, mas esta não tem uma sequência programada. A ideia é que os novos colaboradores aprendam observando outros funcionários mais experientes a executarem a suas tarefas. Este tipo de formação mostra-se menos eficiente em relação ao tipo de formação planeada [9].

Embora a formação não planeada seja menos eficiente que a formação planeada, é a que acontece com mais frequência nas organizações. Hoje em dia existe em cada vez mais empresas a necessidade de recrutar novos funcionários e de criar novos postos de trabalho e novos procedimentos o mais rápido possível. Portanto cada vez mais ter um plano de formação bem estruturado, torna-se essencial para reduzir o tempo de adaptação dos funcionários a novas ferramentas ou procedimentos [9].

Relativamente ao surgimento da formação em contexto de trabalho, a formação planeada apareceu da necessidade dos Estados Unidos terem de produzir estaleiros para a sua participação na primeira Guerra Mundial. Nessa altura, Charles R. Allen propôs um método composto por quatro fases, para formar os seus trabalhadores [9]:

1) Fase de preparação: apresentação ao funcionário o que devia fazer;
2) Fase de apresentação: explicação ao funcionário do que se devia fazer e porque o devia fazer;
3) Fase de aplicação: o funcionário tenta fazer a tarefa;
4) Fase de inspeção: acompanhar o progresso do funcionário, dizendo-lhe se o que ele faz está bem ou não, dando-lhe conselhos sobre a melhor forma de fazer o trabalho.

Já na segunda Guerra Mundial o esquema anterior passou de quatro passos para sete [9]:

1) Mostrar como se realiza a tarefa;
2) Rever os pontos-chave;
3) Permitir que os formandos observem um especialista a executar a tarefa;
4) Permitir aos formandos a execução da tarefa dividida em tarefas mais simples;
5) Orientar os formandos na realização da tarefa completa;

6) Deixar que os formandos realizem a tarefa completa sem qualquer tipo de monitorização;
7) Terminar o treino e deixar os formandos realizarem a tarefa de uma forma autónoma.

Ainda hoje é utilizado este método em muitas empresas industriais com algumas adaptações.
Abordando as dificuldades existentes relativamente à implementação de formação planeada nas empresas, pode-se dizer que é em muitos casos devido aos responsáveis pelas empresas não saberem a diferença entre formação planeada e formação não planeada. Ou mesmo porque simplesmente acham que a formação na sua empresa não é necessária ou não existe tempo para a realizar. Assim a solução para a entrada de novos funcionários para a empresa é simplesmente coloca-los a acompanhar os funcionários mais experientes, de modo a que os novos funcionários possam observar e aprender. Ou simplesmente colocam os novos funcionários no posto de trabalho, obrigando a que estes sozinhos encontrarem soluções para os problemas que vão surgindo. Em muitas empresas a falta de tempo é a grande desculpa para o facto de não existir uma metodologia de formação planeada. No entanto se houvesse uma metodologia planeada corretamente, a suposta falta de tempo poderia ser melhorada.
De uma forma muito geral, o objetivo principal da formação em contexto de trabalho é fazer com que os funcionários das empresas tenham o melhor desempenho possível na realização das suas tarefas. Este desempenho depende de quatro fatores determinantes (**Figura 2.2**).

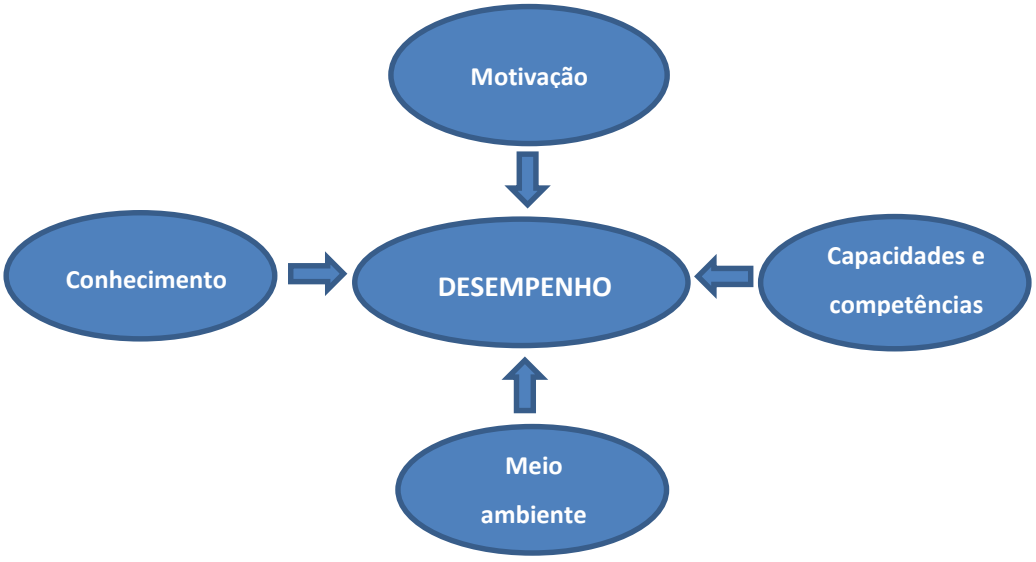

Figura 2.2 - Variáveis do desempenho

O desempenho de um funcionário não depende só de uma variável, como por exemplo dos seus conhecimentos. Para um funcionário conseguir executar as suas tarefas com o melhor desempenho possível, existem outras variáveis que vão influenciar, como podemos ver na **Figura 2.2**. Nesta figura podemos ver que o desempenho depende de quatro variáveis diferentes. Portanto as formações não têm unicamente o objetivo de dar conhecimento técnico ao funcionário. Embora o foco principal da formação seja ensinar ao formando os aspetos técnicos, esta pode construir relações pessoais durante a formação, que vão melhor o seu ambiente de trabalho. É ainda possível que ao perceber as questões técnicas e a sua utilidade o funcionário fique mais motivado. Sendo assim, é importante ter a consciência do que é necessário planear para a formação para que se consiga fazer com que o funcionário tenha o melhor desempenho possível.

2.1.3 Organização da Formação

A formação ainda é vista por muitas organizações como um custo sem retorno. No entanto, a formação prática em contexto de trabalho tem vindo a revelar-se como sendo responsável por trazer aos funcionários um bom desempenho profissional [10].

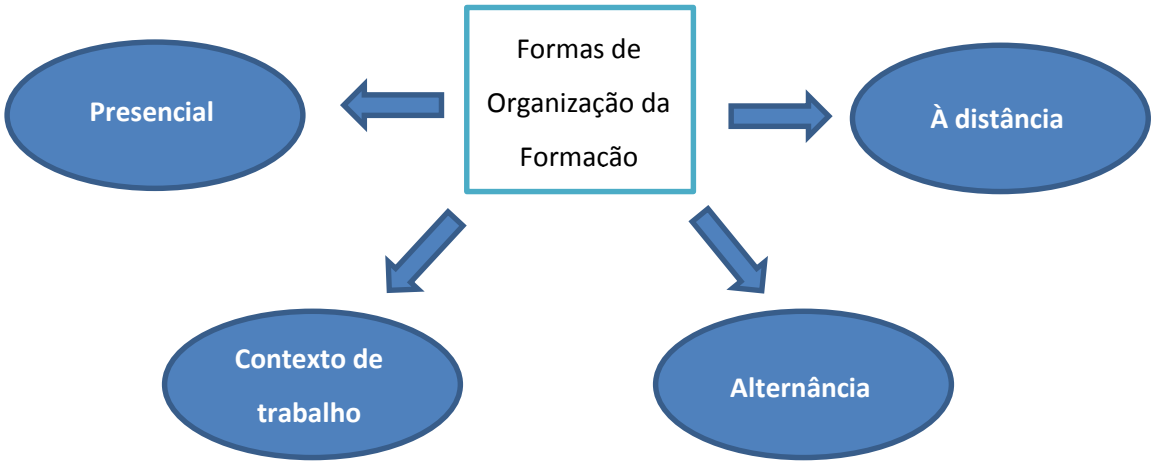

Figura 2.3 - Organização de Formação

Baseando-nos na **Figura 2.3**, as formações podem ser organizadas em quatro tipos diferentes [10]:

- **Formação presencial:** formações realizadas numa sala, com a presença de um formador de modo a facilitar as interações entre os formandos e o formador;
- **Formação em contexto de trabalho:** atividade desenvolvida no próprio local de trabalho ou num local similar de modo a simular o ambiente de real de trabalho;
- **Formação em alternância:** formações onde se vai alternando entre o tipo de formação presencial com o tipo de formação em contexto de trabalho;

- **Formação à distância:** também conhecido com *e-learning*, que se define pela flexibilidade de quando é feita e onde, pois este tipo de formação é realizada através de um portal na Internet. Este tipo tem pouca ou mesmo nenhuma intervenção do formador. Para os casos em que este tipo se alterna com o tipo presencial o nome passa a ser *b-learning*.

Uma vez que existe um grande número de situações em que a formação pode ser utilizada, é difícil existirem modelos pré-definidos. No fundo, cada caso é um caso e para preparar uma formação é necessário estabelecer bem os seus objetivos. Então, para ajudar na sua preparação relativamente à sua finalidade e objetivos tendo em consideração as necessidades de uma organização, podem ser feitas algumas questões às quais cada organização deve tentar responder [9]:

- Qual é a população-alvo?
- Porque deve ser aplicada?
- Quando deve ser aplicada?
- Onde deve ser aplicada?
- Como integrar o plano da formação no plano estratégico da empresa?
- Como pode a organização dar a conhecer aos novos colaboradores o modelo de funcionário pretendido?

Os formandos dentro de uma organização podem estar associados a muitas funções. Assim, quando são desenhados os programas de formação, não se deve apenas olhar para as necessidades existentes da empresa mas também deve ser considerado o nível em que cada formando se encontra. Outro ponto também importante de referir é o facto de muitas empresas terem a ideia que a formação deve ser dada apenas aos novos funcionários. No entanto a formação deve ser uma prática continua permitindo assim que todos os funcionários se adaptem a novas alterações.

Impacto da Formação no Compromisso do Funcionário com a Empresa

Como já foi dito anteriormente, a formação é um aspeto muito importante dentro de uma organização. As empresas que optam por investir na formação dos seus colaboradores, normalmente conseguem obter resultados positivos desse investimento. Ou seja, com funcionários bem formados a produtividade e os respetivos lucros de uma empresa vão subir. Há quem defenda que a formação traz aumentos no volume de negócio e há quem defenda também que a formação faz com que os funcionários fiquem por longos períodos de tempo numa organização [11]. No entanto a formação tem custos para as empresas, são eles diretos (salário do formador, material de formação, entre outros) ou indiretos (custos na produção durante a formação). Mas depois de

a formação terminar é esperado que o rendimento do funcionário no seu posto de trabalho aumente o suficiente para a empresa ver os seus custos retornados. Além da empresa o próprio funcionário sai a ganhar, pois com mais formação o seu desempenho tende a aumentar o que vai promover uma maior progressão de carreira e um aumento salarial.

A formação é uma das várias práticas de recursos humanos que podem ter um impacto considerável sobre o compromisso do empregado.

Formação e Investimento do Empregado

Gary Becker (prémio nobel da economia em 1992) identificou duas formas diferentes de formação: formação geral e formação específica. A formação geral é relativa a aspetos que podem ser úteis para muitos empregadores (como por exemplo ao nível do MS Office, de sistemas de contabilidade, de maquinaria geral). Os cursos superiores podem ser vistos também como formação geral, visto que alunos estão a aprender sobre aspetos que são úteis para muitas empresas. A formação específica é aquela que tem como alvo uma determinada empresa, sendo que todos os conhecimentos não serão úteis para utilizar noutras empresas.

O investimento por parte das empresas em formação geral normalmente não acontece. Pois a empresa que pagou a formação ao funcionário, no final terá de aumentar o seu salário, visto que o funcionário aumentou as suas habilidades. E as empresas que não o fizerem podem ver o seu funcionário a demitir-se e a ir para outra empresa. Pois com os conhecimentos que agora tem, conseguirá um salário melhor. Sendo assim, os custos das formações associados a possíveis aumentos salariais, podem tornar-se insustentáveis para as empresas. Desta forma, são muitas vezes os trabalhadores que têm de suportar os custos de formação geral.

Assim, quanto mais especificas são as competências necessárias, mais fácil é para uma empresa pagar a formação, pois não existem outras empresas onde os trabalhadores podem utilizar os conhecimentos adquiridos. Então essa formação vai fazer com que o rendimento produtivo do funcionário seja maior para a empresa que pagou a formação, mas se mantenha igual para as outras empresas. Assim a empresa pode pagar a formação e não ter de aumentar imediatamente o salário do funcionário. O funcionário sabe ainda que a formação o torna importante dentro da empresa e sabe que isso poderá resultar num aumento de salário futuro. Além disso, como o funcionário também dedicou esforço à formação e sabe que se sair da empresa a formação não lhe valerá de nada (colocando-o novamente no mesmo patamar que estava quando entrou na empresa), faz com que o funcionário se mantenha na empresa com mais facilidade.

Formação e Reciprocidade

A formação geral pode revelar-se capaz de criar compromisso, mas apenas pelo facto de fazer com que o funcionário se sinta em divida para com a empresa. Isto é, pode criar compromisso através de reciprocidade. Assim o funcionário sente-se na obrigação de se manter na empresa até pelo menos esta conseguir recuperar o investimento que fez nele.

Formação e Identidade Social

A formação tem um papel importante para a socialização dos funcionários. Na maioria das vezes, os novos funcionários entram para um novo emprego com expectativas bastante positivas. Quando essas expectativas se concretizam, o funcionário passa a identificar-se mais com a empresa o que o faz ficar mais comprometido com ela. No entanto, quando essas expectativas não se concretizam, o funcionário não se identifica com a empresa o que fará com que este não tenha um rendimento tão bom e consequentemente, não fique comprometido com a empresa.

2.1.4 Consequências da Falta de Formação

É de conhecimento comum que as empresas dependem da qualidade do trabalho dos seus colaboradores. Portanto, todas as empresas (principalmente as que tem métodos de trabalho com alguma complexidade) devem ter em consideração a formação dos seus colaboradores, para que deste modo eles possam corresponder da melhor forma. Quando isso não acontece os funcionários ficam inseguros relativamente às tarefas que devem fazer. E isso obviamente traz problemas relacionados com a eficiência nas tarefas dos funcionários. A falta de entendimento entre os funcionários começa a aparecer pela falta de clareza nos requisitos. E com isso outros problemas surgem, tais como a desmotivação dos funcionários, o desentendimento entre eles e muitos outros que vão ajudar a chegar ao caos. E quem perde mais com o surgimento destes problemas é própria empresa [12].

Mais tarde ou mais cedo, os funcionários que estão insatisfeitos acabam por sair. E para ocupar o cargo do funcionário que saiu, outro terá de ser contratado. Mas se o método de não dar formação se mantiver, esse novo empregado vai também ficar insatisfeito e, por conseguinte, sair da empresa. Assim, enquanto a empresa não mudar o seu método, este ciclo dificilmente termina. O que vai fazer com que a empresa gaste ainda mais dinheiro do que gastaria, se tivesse um método de formação para os seus colaboradores. Além disso, é de conhecimento comum que a produtividade de um funcionário motivado é relativamente maior do que a de um desmotivado [12].

2.1.5 Benefícios na Formação

De uma forma resumida os objetivos da formação são a melhoria dos conhecimentos, das capacidades e das atitudes em relação às tarefas que são realizadas por um determinado funcionário. A formação consegue muitas das vezes trazer motivação, tanto a curto como a longo prazo, para os funcionários da empresa. Abaixo temos alguns pontos positivos da implementação de formação [13]:

- Aumento de confiança: os empregados que recebem formação vão ter obviamente mais confiança na realização das suas tarefas;
- Menor custo de produção: como os funcionários são treinados, vão saber utilizar melhor o material e o equipamento necessário para as suas funções. Assim é reduzido o desperdício e também é diminuída a probabilidade de danificar algum equipamento.
- Baixa rotatividade de pessoal: para além de trazer uma sensação de segurança, a formação faz com que os funcionários não queiram sair, simplesmente porque sentem um compromisso para com a empresa. E também pelo facto de muitas vezes não conseguirem aplicar os conhecimentos já adquiridos nas formações noutras empresas.
- Gestão da mudança: a formação providencia as capacidades necessárias para ser feito um ajuste a novas situações;
- Proporciona reconhecimento sendo reforçada a responsabilidade, o que poderá vir a promover uma promoção na carreira e um aumento salarial;
- Ajuda a ter uma equipa de funcionários mais disponível e com mais qualidade.

2.2 Soluções de Formação

Neste subcapítulo são abordados três soluções que têm como objetivo auxiliar novos utilizadores de um dado sistema a perceber como realizar determinadas tarefas.

2.2.1 Toonimo

Toonimo é um sistema *Software as a Service (*SaaS), que permite às empresas apresentar as funcionalidades da sua plataforma através de guias audiovisuais personalizados. Este produto combina de uma forma sincronizada voz com marcas (círculos, setas ou outras imagens gráficas) que vão sendo feitas na página, salientando as zonas onde é preciso que o utilizador se concentre. Apesar de se assemelhar a um vídeo, permite que o utilizador interaja com a página que está a ser apresentada. Desta forma, é como se os utilizadores tivessem um instrutor ao lado deles a dar indicações de como fazer uma determinada tarefa.

Como funciona:

De modo a entender melhor, vamos utilizar um exemplo dado pela própria empresa. Imaginemos uma página (**Figura 2.4**) que apresenta os custos do serviço de uma empresa a um cliente seu. No entanto, nem todas as pessoas percebem os detalhes dos custos de uma forma detalhada. Dessa forma quando um utilizador entra na página onde são listados os detalhes da conta, irá ser apresentada a hipótese de ser feita uma explicação sobre os detalhes dessa página. Se o utilizador pretender ver uma explicação, uma voz começa a definir cada um dos detalhes da página, e ao longo da explicação vai "sublinhando" os elementos que estão a ser abordados pela voz.

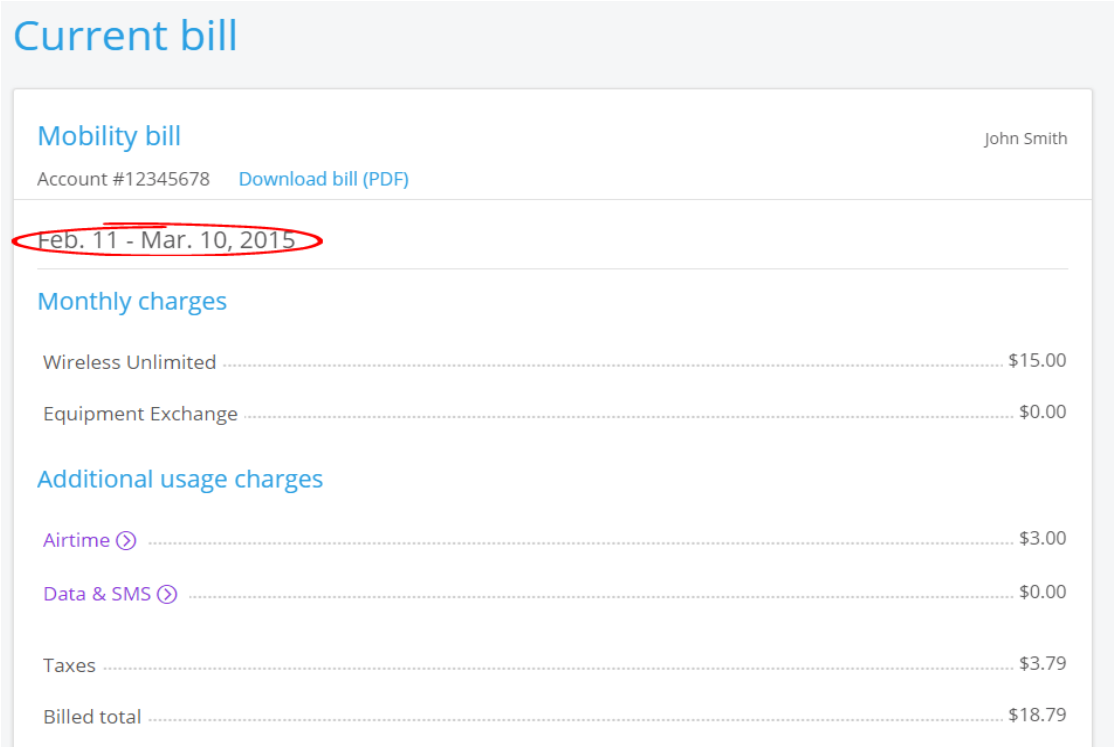

Figura 2.4 - Exemplo Toonimo

Na **Figura 2.4**, no momento em que foi retirada esta imagem, a voz está a dizer que a conta em questão é relativa ao período de 11 de fevereiro a 10 de março de 2015. Vemos que o elemento que está marcado é onde é apresentado o período relativo à conta em questão.

Este projeto não disponibiliza um período de teste e o preço praticado é a partir de 290 dólares por mês [14]. O seu modelo de negócio é o modelo de subscrição.

2.2.2 Whatfix

Whatfix é uma *startup* baseada em Bangalore, India, que tem como objetivo criar guias para ajudar utilizadores a navegar em uma plataforma Web. Foi fundada por Khadim Batti e Vara Kumar em

abril de 2013. O seu produto permite a criação por parte de empresas, de material para formar utilizadores de uma dada plataforma. Um exemplo pode ser a criação de tutoriais interativos baseados em *tip-ballons*, que são associados a elementos na página Web onde o utilizador interage com o sistema [15]. Existe ainda a possibilidade de criar demonstrações de como realizar uma determinada tarefa através de slides ou através de vídeo.

Este projeto oferece um sistema que possibilita que uma dada organização consiga criar fluxos de ajuda para serem disponibilizados aos seus utilizadores. A criação destes fluxos pode ser criada por pessoas que não tenham conhecimento técnico acerca do desenvolvimento do sistema, apenas devem conhecer o sistema do ponto de vista do utilizador. Para a criação desses fluxos é utilizada uma extensão para o *browser* (existente para o Chrome e Firefox), permitindo assim a sua criação, clicando nos elementos da página necessários para a realização da tarefa. À medida que é feita a escolha dos elementos na página HTML, é feita uma rápida configuração relativa a cada um dos elementos. Como por exemplo o texto que será apresentado ao utilizador relativamente ao elemento.

É possível a criação de ajudas com base num determinado contexto. Isto é, uma plataforma pode ter sempre acessível ao utilizador novo no sistema, uma secção de *Frequently Asked Questions* (FAQ) interativas, onde este pode esclarecer dúvidas sobre uma determinada tarefa. Basta para isso aceder a essa secção e escolher um guia que o ajude a completar a sua tarefa.

Como funciona:

De forma a dar um exemplo deste produto, será utilizado o exemplo do SIJ. A tarefa é relativa aos primeiros passos da adição de um novo arguido a um processo.

Depois de ter a extensão para o *browser* instalada, deve-se aceder à página onde se quer configurar o material de apoio. De seguida deve-se clicar na extensão do whatfix para que seja apresentada a *modal* que vemos na **Figura 2.5**.

Figura 2.5 - Modal início whatfix

Nesta *modal* deve-se adicionar uma pequena descrição relativa ao fluxo que vai ser criado. Para começar a adicionar passos ao fluxo, é necessário clicar no botão laranja "+step". A *modal* irá então desaparecer, ficando os elementos da página visíveis. Aqui deve ser feito o clique no elemento que quer associar a ajuda, ou seja, o elemento que deve ser clicado para executar a tarefa. Sendo este selecionado, será apresentada novamente a *modal* (**Figura 2.6**) mas agora para ser configurado este passo.

Figura 2.6 - Editar processo

Nesta *modal,* deve ser inserido um texto que descreva a tarefa. Podem ainda ser definidos outros aspetos, tais como a posição que onde irá aparecer o *tip-ballon* relativamente ao elemento, se esta tarefa é ou não obrigatória, entre outros. Quando o passo estiver pronto, deve-se clicar em "*add step*" para terminar a configuração deste passo.

A seguir, deve-se ir para a página relativa ao próximo passo, **Figura 2.7**. Aqui, deve ser feito o mesmo que foi feito anteriormente, ou seja selecionar o elemento necessário para continuar a tarefa, de modo a poder associar-lhe um *tip-ballon*.

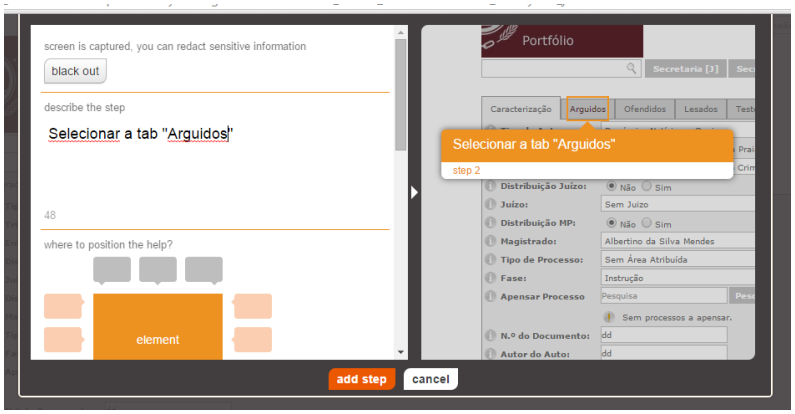

Figura 2.7 - Clica tab arguidos

O mesmo deve ser feito para todas as interações necessárias para a realização da tarefa. Assim, quem realiza a tarefa vai ter ajuda ao longo de todos os passos.

Olhando agora do ponto de vista do utilizador novo no sistema que vai usufruir destes *tip-ballons.*, quando este estiver para adicionar um novo arguido ao processo, vai ter a ajuda dos *tip-ballons*, como podemos ver na **Figura 2.8**, que indica que é necessário clicar no "Editar processo".

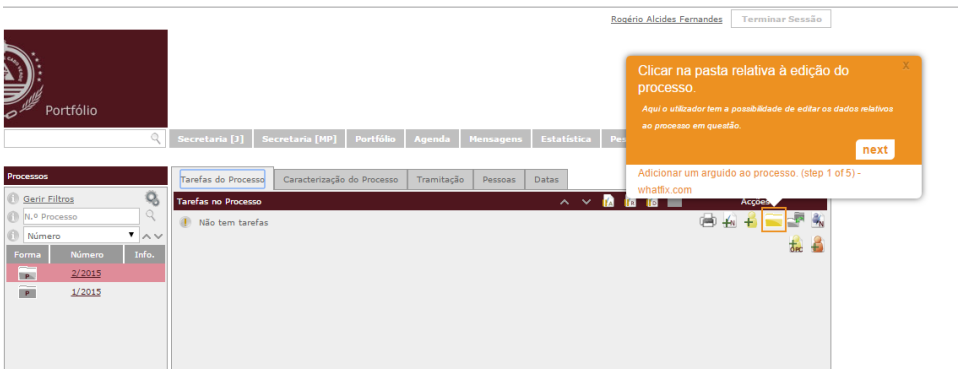

Figura 2.8 - *Tip-Ballon* editar processo

Podemos ver também o *tip-ballon* (**Figura 2.9**) apresentado na altura em que deve ser escolhida o separador “Arguido”.

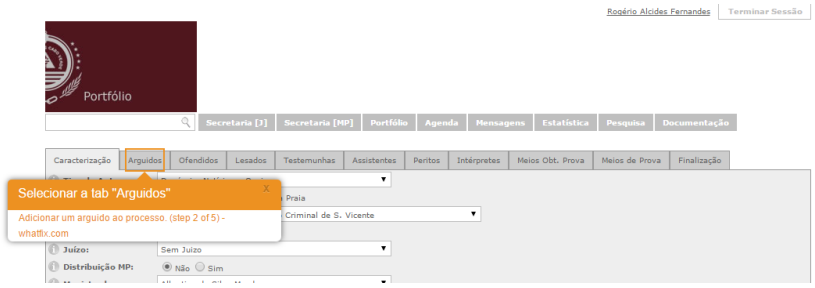

Figura 2.9 - *Tip-Ballon* tab arguido

Os novos utilizadores do sistema não têm apenas a hipóteses de usufruir da funcionalidade relativa ao surgimento dos *tip-ballons* à medida que navegam no sistema. Depois de serem feitas as configurações relativas aos *tip-ballons*, fica também disponível em formato de vídeo, em formato de artigo e em *slides* como fazer uma determinada tarefa.

Com whatfix é possível providenciar aos utilizadores um *widget* (que está sempre presente na página) para que estes possam em qualquer altura, consultar e usufruir de todas as opções de ajuda.

Relativamente ao modelo de negócio o whatfix segue o modelo *freemium* [14]. Ou seja, disponibiliza algumas funcionalidades grátis, no entanto, se os utilizadores quiserem ter acesso a mais funcionalidades, estas serão cobradas.

2.2.3 WalkMe

WalkMe foi fundada em 2011 lançando a sua plataforma em abril de 2012. Segundo a própria empresa, tinham a missão de melhorar a experiência dos utilizadores em plataformas *online* a nível de usabilidade [16].

Alguns dos seus clientes mais conhecidos são a MasterCard, a PayPal, a Cisco, a Pandora e a Virgin.

Este projeto tem como objetivo permitir que os proprietários de uma dada plataforma possam oferecer aos seus utilizadores um guia para os ajudar a navegar no seu sistema, sem estes ficarem confusos relativamente à realização de uma determinada tarefa. Como diz a própria WalkMe, este sistema pode ser comparado a um GPS, que em vez de dar as direções para uma determinada zona, orienta o utilizador em cada um dos passos necessários para a realização de uma determinada

tarefa. Assim, com recurso a *tip-ballons* apresentados na altura adequada, vão sendo transmitidas informações para que o utilizador saiba o que deve fazer naquele momento.

Este sistema tem dois aspetos que se destacam. O primeiro é o facto de permitir a criação de segmentos de utilizadores. Assim, com esta funcionalidade torna-se possível criar guias de orientação para tipos de utilizadores diferentes. Este aspeto é importante para casos em que a maneira de realizar uma determinada tarefa depende do tipo do utilizador. Vamos imaginar a situação em que o objetivo é orientar o utilizador a realizar o pagamento dos itens que tem no seu carrinho de compras de uma plataforma de vendas *online*. Para esta tarefa podemos ter dois tipos de utilizadores: utilizadores com itens no carrinho de compras e utilizadores sem itens no carrinho de compras. Assim, para o utilizador que tem itens no carrinho de compras, será apresentado um *tip-ballon* que indica onde este deve clicar para proceder ao pagamento dos itens que se encontram no seu carrinho. Já para o utilizador que não tem nenhum item no seu carrinho, será apresentado em primeiro lugar um *tip-ballon* que visa orientar o utilizador a adicionar um item ao seu carrinho, e apenas depois será apresentado o *tip-ballon* para orientar o utilizador a proceder ao pagamento.

O segundo aspeto é a capacidade que este sistema oferece relativamente à validação dos passos realizados pelo utilizador. Vamos imaginar um guia de orientação para a realização de uma determinada tarefa, que é constituída por vários passos. Se num determinado passo o utilizador não faz o que é suposto fazer, o sistema pode reencaminhar o utilizador para o sítio onde este errou, de modo a que ele tenha a possibilidade de fazer a ação corretamente, evitando assim que seja obrigado a recomeçar a tarefa de início. Tomamos por exemplo uma situação onde um utilizador deve carregar num determinado botão para ser reencaminhado para uma determinada página. No entanto ele engana-se e carrega noutro botão que o reencaminha para outra página. Acontecendo isto, o sistema volta a reencaminhar o utilizador para a página onde este clicou no botão errado, de modo a que ele agora possa clicar no botão correto sendo assim reencaminhado para a página certa.

Para utilizar o sistema WalkMe, relativamente à criação de guias de orientação, é necessário adicionar uma extensão ao *browser* que ainda existe apenas para o Mozilla Firefox.

Como funciona:

De forma a se perceber melhor este sistema, é utilizado o mesmo exemplo já usado para demonstrar o sistema whatfix. O exemplo é relativo aos primeiros passos necessários para a adição de um arguido a um processo já existente. De modo a se tornar menos confuso, são utilizadas apenas as funcionalidades básicas do WalkMe.

Para começar a realizar guias, é necessário instalar a extensão “walkme” no Mozilla Firefox. A partir desse momento, quando temos a extensão ativa, é apresentado do lado esquerdo do ecrã o editor (**Figura 2.10**) que permite a criação de novos guias.

Para criar novos guias, o utilizador deve clicar no botão redondo azul com o sinal de “+”. Será então apresentado uma caixa de texto para o utilizador introduzir o nome para o guia em questão.

Depois de criado o guia, o utilizador pode clicar em cima dele, abrindo assim todos os passos associados a este guia.

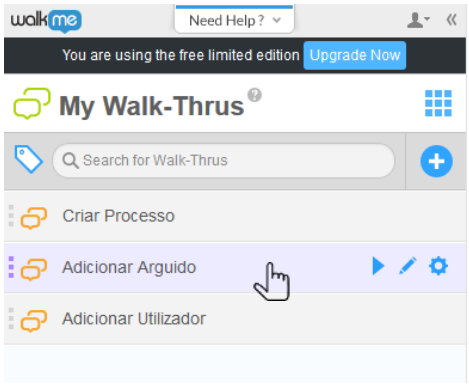

Figura 2.10 - WalkMe Walk- thrus

Como o guia acabou de ser criado, não vão existir passos associados a ele. Para adicionar novos passos é necessário clicar no botão redondo azul com o sinal de “+”. Aí o utilizador deve selecionar o elemento na página em que quer associar a ajuda. Depois de clicado irá aparecer uma *pop-up* para ser configurado o que será apresentado ao utilizador novo no sistema. Neste exemplo, deve-se clicar na pasta relativa à edição do processo e será então apresentada a *pop-up*, como podemos ver na **Figura 2.11**. Aqui pode ser configurada a posição do *tip-ballon*, bem como o que fará com que este passo seja dado como realizado, de modo a que o *tip-ballon* desapareça. No entanto, o ponto mais importante é a introdução do texto que será apresentado no *tip-ballon* quando a tarefa estiver a ser realizada. Para terminar deve-se clicar em “*Done*”.

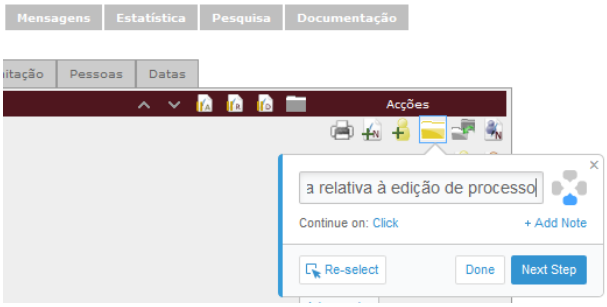

Figura 2.11 - WalkMe primeiro passo

De seguida, o mesmo deve ser feito para a criação de todos os passos necessários. Deve-se então ir para a página onde deseja adicionar a ajuda e clicar no botão redondo azul com o sinal de "+". Depois deve-se selecionar o elemento onde se quer associar a ajuda, e configurar a *pop-up* que irá aparecer bem como introduzir o texto que será apresentado no *tip-ballon*.

À medida que os passos vão sendo adicionados ao guia, estes vão sendo listados. Podemos ver na **Figura 2.13** todos os passos associados ao guia "Adicionar Arguido".

Figura 2.13 - WalkMe passos do guia "Adicionar Arguido"

Exemplificando agora do ponto de vista do utilizador novo no sistema, que terá a ajuda na realização das tarefas. Quando este estiver na página onde começa o guia, é-lhe apresentado uma *pop-up* (**Figura 2.12**) que indica onde é que deve clicar para realizar o primeiro passo da tarefa "Adicionar Arguido".

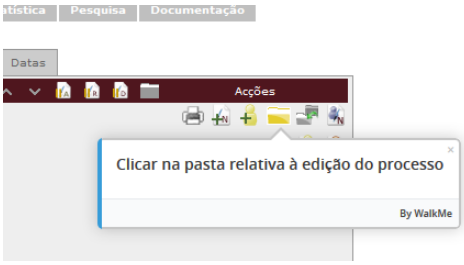

Figura 2.12 - WalkMe execução do passo 1

À medida que o utilizador vai concretizando os seus passos, novos *tip-ballons* são apresentados para indicar constantemente o que deve o utilizador fazer para realizar a sua tarefa.

Relativamente ao seu modelo de negócio, o WalkMe segue também o modelo *freemium* [14].

2.2.4 Comparação

À medida que os sistemas foram sendo estudados, cada vez mais se tornavam visíveis as mais-valias que estes podem trazer. Especialmente quando estamos perante um sistema complexo e existe a necessidade de transmitir aos utilizadores como se devem fazer determinadas tarefas.

	Guias de ajuda	Validação	Guia de voz	Avaliação grátis
Toonimo	✗	✗	✓	✗
Whatfix	✓	✗	✗	✓
WalkMe	✓	✓	✗	✓

Tabela 2 - Comparação de soluções

O sistema Toonimo, difere dos outros dois relativamente ao facto de este explicar o sistema com base em voz e em marcas na própria página. Transmite a sensação de ter alguém sentado ao lado a explicar o sistema e a apontar para os sítios na página que importam.

Relativamente ao whatfix e WalkMe, estes são semelhantes no que diz respeito ao facto de permitirem a criação de guias. Estes vão conduzindo o utilizador através do sistema com recurso *pop-up's* que são apresentadas na altura adequada, transmitindo ao utilizador o que este deve fazer. Cada um dos sistemas tem as suas vantagens e as suas desvantagens. Uma das vantagens que o whatfix tem, é o facto de poder transformar os guias já definidos em vídeo, em slides ou em documentos. Podendo assim o utilizador consultar previamente como se faz determinada tarefa. Relativamente às vantagens do sistema WalkMe, destaca-se o facto de ser possível associar validações a cada passo, conseguindo assim garantir que o utilizador faça a tarefa de forma correta.

No entanto, nenhuma destas opções consegue satisfazer as necessidades para a formação de utilizadores do Sistema de Informação da Justiça. De acordo com os objetivos, existe a necessidade de fazer validações ao nível dos dados que são inseridos pelos formandos. Estas validações implicam uma consulta direta na base de dados, de forma a validar que os dados inseridos estão em concordância com os outros dados relativos ao processo em questão. Neste ponto, o WalkMe é semelhante com pretendido.

Outra necessidade que os sistemas abordados anteriormente não conseguem satisfazer, é o facto de ser necessária a simulação de tarefas relativamente a outros utilizadores. Como existem tarefas que só podem ser realizadas depois de outras, iria acontecer o caso de alguns utilizadores terem de esperar pela realização de tarefas da competência de outros. Assim é necessário poder simular

tarefas, para que não haja utilizadores à espera que outros façam as tarefas que lhes competem. A possibilidade de simular tarefas permite também que formandos possam iniciar uma determinada formação, estando sozinhos. Com a escolha de um determinado *template* ou formação*,* as tarefas relativas a outros tipos de utilizadores, serão simuladas pelo sistema. Desta forma, o utilizador consegue fazer qualquer tarefa sem depender de outros utilizadores.

3 Especificação

Neste capítulo é feita uma especificação relativa ao desenvolvimento do sistema. No primeiro subtópico são apresentados os requisitos para o desenvolvimento do sistema. No segundo será abordada a arquitetura do sistema. E por último, no terceiro subtópico será descrito o modelo de dados.

3.1 Requisitos

Um dos primeiros passos quando se planeia desenvolver um sistema de informação deve ser um levantamento de requisitos, para se perceber quais as necessidades do sistema que se virá a desenvolver. Esta atividade é responsável por definir os serviços que um sistema deve ter e quais as restrições, estabelecendo assim o que deve o sistema fazer em vez de definir como será feito. As pessoas envolvidas no processo de levantamento de requisitos são os clientes ou os utilizadores, que vão requerer as necessidades do sistema. São também os analistas, que são responsáveis por extrair as necessidades requeridas pelos clientes ou utilizadores e fazer a sua validação, de forma a fazer uma descrição detalhada das necessidades. Por último temos as pessoas que vão desenvolver o sistema a partir da descrição detalhada realizada pelos analistas. Esta prática tende a antecipar o surgimento de desentendimentos relativos aos requisitos necessários, trazendo assim uma melhor eficiência no seu desenvolvimento.

3.1.1 Casos de Uso

O objetivo de um diagrama de casos de uso é apresentar as principais funcionalidades do sistema do ponto de vista do utilizador, mostrando então através de um diagrama o que um utilizador pode fazer quando utiliza o sistema.

A seguir são apresentados três casos de uso que mostram de uma forma mais geral as funcionalidades propostas para o sistema. Embora não esteja representado nos casos de uso, todas as funcionalidades implicam a autenticação no sistema.

3.1.2 Caso de Uso 1: Gestão de *Template*

Aqui são apresentadas as funcionalidades relativamente à gestão dos *templates* com passos para os formandos executarem na formação. Todas as funcionalidades apresentadas neste diagrama são somente permitidas aos utilizadores com o *role* administrador.

Atores:

- Administrador

Casos de uso:

- Criar passo – criar um dos vários passos que constituem um *template*.
- Editar passo – editar um passo já criado.
- Remover passo – remover um passo já criado.
- Criar template – criar um *template*. Aquando da sua criação o utilizador pode adicionar documentos para mais tarde serem descarregados. A criação de um *template* implica a criação de vários passos.
- Adicionar documentos – quando o utilizador está a criar o *template* pode adicionar documento que ficam associados a esse mesmo *template*.
- Ver templates – ver uma lista com todos os *templates* criados. Aqui o utilizador pode também editar os documentos que estão associados a este template.
- Editar documentos – o utilizador pode editar os documentos associados a um determinado *template*.
- Editar template – o utilizador pode editar um *template* já criado. Para conseguir realizar esta tarefa é necessário, em primeiro lugar, listar todos os *templates*.
- Remover template – o utilizador pode remover um *template* já criado. Esta tarefa implica também que sejam listados todos os *templates* para o utilizador poder escolher qual quer remover.

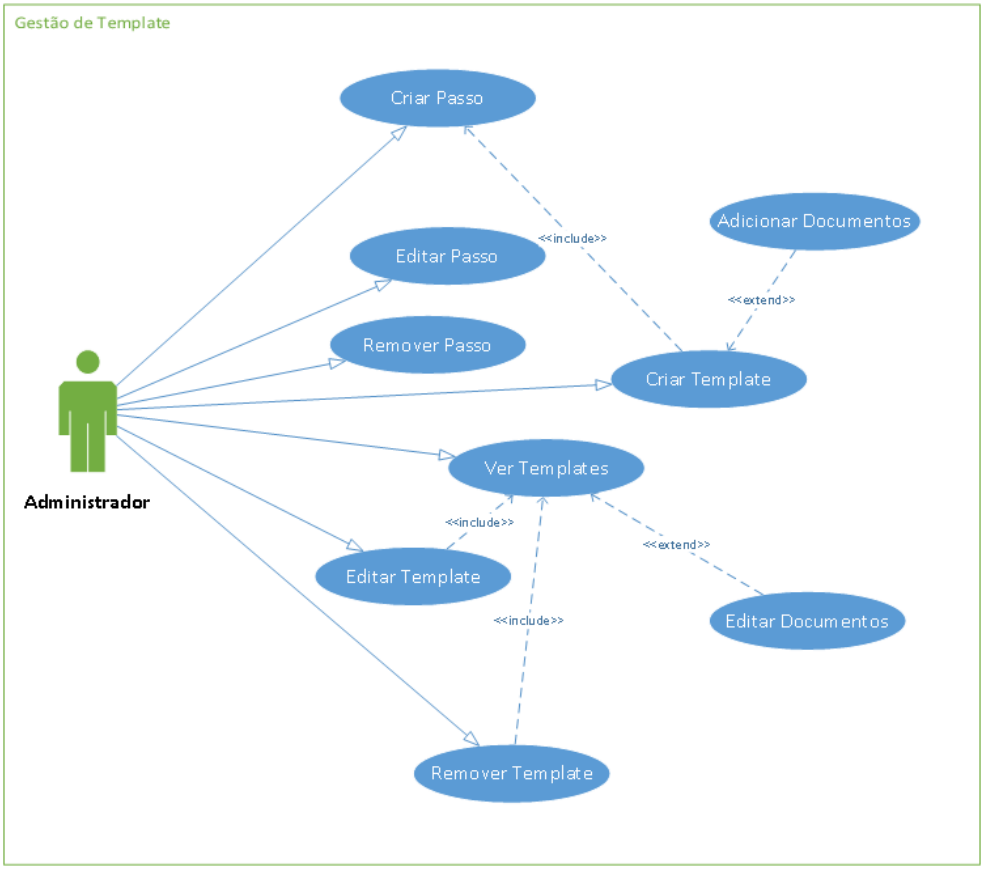

Figura 3.1 - Caso de uso gestão de template

3.1.3 Caso de Uso 2: Gestão de Configuração de Perfis

As funcionalidades apresentadas neste diagrama são relativas às tarefas de criação e configuração de perfis, tendo como base os *template* já criados.

Atores:

- Administrador;
- Formando.

Casos de uso:

- Listar Templates – listar todos os *templates* já criados.
- Criar configuração – para um determinado *template*, o utilizador deve configurar os *roles* que vão fazer cada um dos passos do *template* e deve indicar também quais os passos que devem ser realizados pelos formandos e quais são os passos para serem simulados pelo sistema.

- Configurar passos – para cada passo o utilizador deve dizer qual o role que deve realizar esse passo e se é um formando ou o sistema a realizar o passo.
- Editar configuração – atualizar uma dada configuração.
- Remover configuração – remover uma dada configuração.
- Listar configurações – listar todas as configurações existentes no sistema.

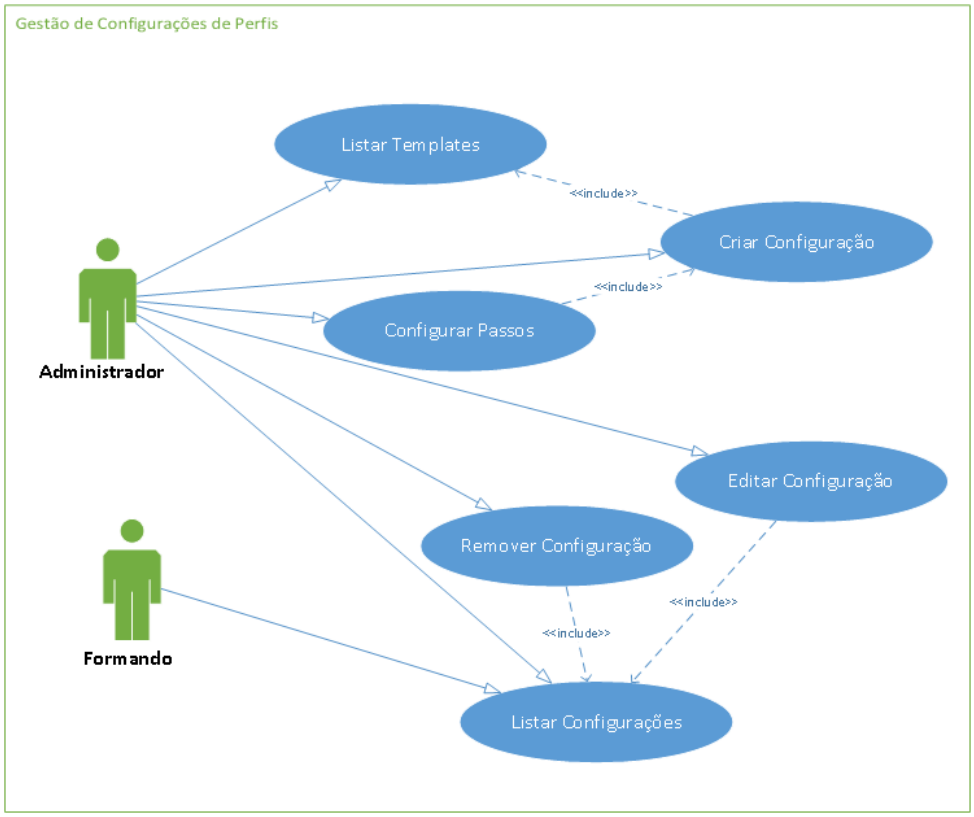

Figura 3.2 - Caso de uso gestão de configuração de perfis

3.1.4 Caso de Uso 3: Arranque e Monitorização da Formação

Este diagrama apresenta as funcionalidades principais relativas ao arranque e monitorização de formações por parte do administrador bem como pelo formando.

Atores:

- Administrador;
- Formando.

Casos de uso:

- Ver formações começadas - ver uma lista com todas as formações que já começaram.
- Listar concretizações – ver uma lista com todas as configurações realizadas com base nos *templates*.
- Definir utilizadores – definir quais são os utilizadores que vão realmente fazer cada passo.
- Arrancar formação – começar uma formação com base numa configuração de perfis já realizada.
- Monitorização de passos – ver o estado de cada passo associado a um *template* para uma determinada formação. Aqui o utilizador pode consultar formações que já acabaram bem como formações que ainda estão a decorrer.
- Inserir número de processo – quando o primeiro passo é suposto ser feito pelo formando, este deve fazer a criação do processo no SIJ e ver o respetivo número de processo para introduzir no sistema da formação. De modo a que este consiga saber qual o processo a monitorizar.
- Descarregar documentos – quando o utilizador está a ver o estado da formação, pode descarregar os documentos que estão associados à formação em questão.
- Terminar formação – quando as formações estão a decorrer o utilizador pode terminar a formação.
- Reiniciar formação – o utilizador pode reiniciar uma determinada formação.

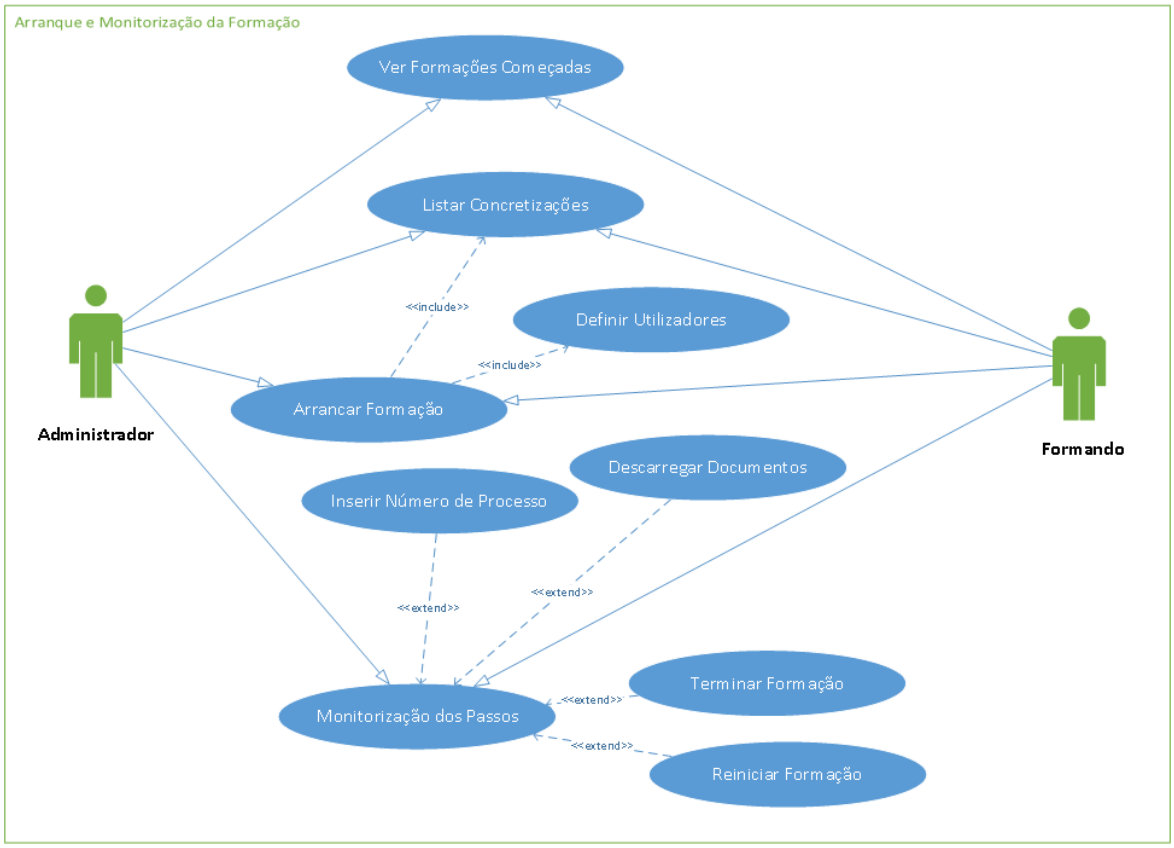

Figura 3.3 - Caso de uso arranque e monitorização da formação

3.2 *Arquitetura*

O sistema a que se propõe este trabalho está relacionado com outros sistemas. Dessa forma, é necessário ter bem definido a sua relação com os outros sistemas de modo a evitar problemas no futuro. Assim, a solução a que se propõe esta dissertação é relativa à parte de criação, configuração e monitorização de uma dada formação. Portanto é necessário ter em consideração a comunicação com o SIJ, bem como a sua relação com a parte do sistema da formação responsável pela validação dos passos realizados pelos utilizadores e pela simulação de tarefas.

Na **Figura 3.4** podemos ver a existência de dois grupos principais. Um dos grupos é relativo ao SIJ e outro é relativo ao sistema da formação. Por sua vez o sistema da formação é também dividido em dois. A parte relativa à criação, configuração e monitorização da formação e a parte relativa à validação das tarefas feitas pelos formandos e simulação de ações. Podemos ver ainda na **Figura 3.4** que as duas partes do sistema da formação partilham a mesma base de dados. Assim, a partir da aplicação Web, o utilizador pode fazer a criação dos *templates*, criar várias configurações acerca dos perfis dos utilizadores e começar formações, sendo guardados os dados relativos a esses pontos

na base de dados. Deste modo, na respetiva altura, o sistema de validação e simulação, pode consultar a base de dados de forma a saber o que tem de validar e o que tem de simular.

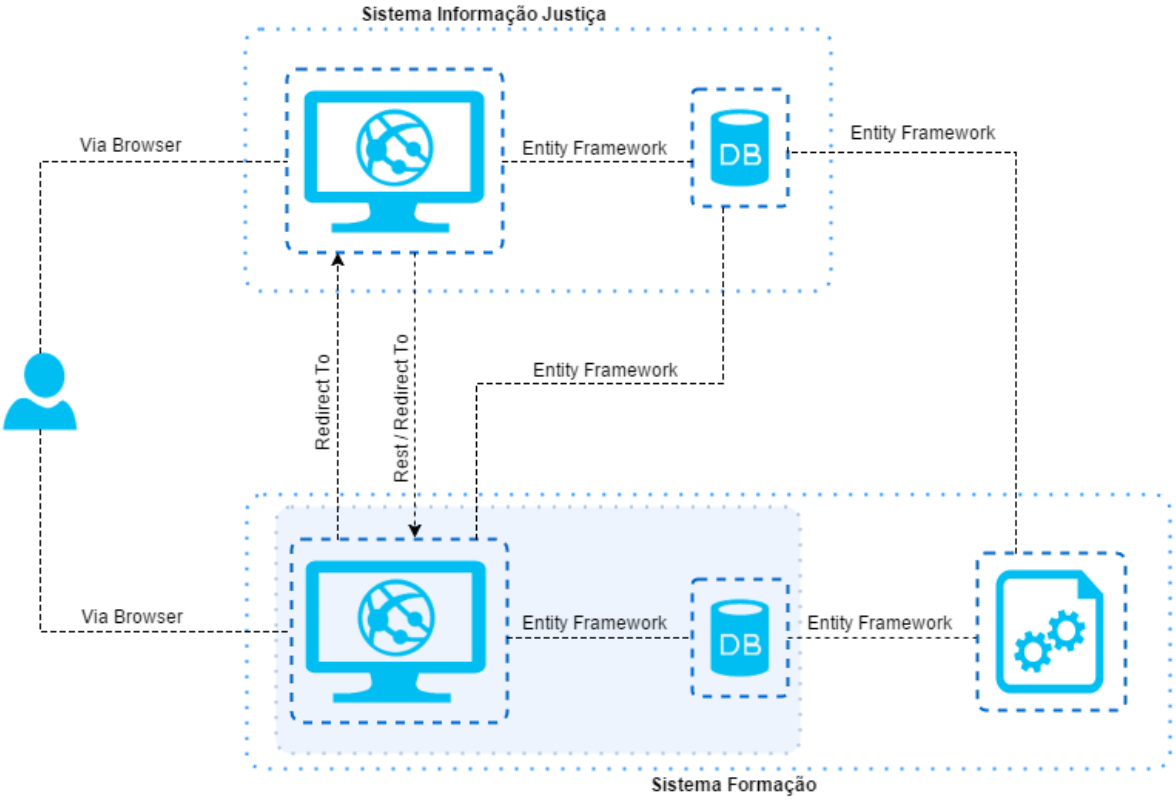

Figura 3.4 - Arquitetura

Na **Figura 3.4** vemos ainda uma relação do sistema de validação e simulação com a base de dados do SIJ. Esta relação faz com que se consiga monitorizar as tabelas necessárias na base de dados do SIJ, sabendo assim se o formando fez o que efetivamente deveria fazer. Para isso é necessário a comparação dos dados que o utilizador inseriu com os dados que foram definidos na criação do *template* respetivo à formação em questão. A outra finalidade desta relação é o sistema de validação e simulação, simular a tarefa de um formando, inserindo os dados que foram definidos aquando a criação do *template* nas tabelas corretas.

Vemos ainda uma relação entre a aplicação Web do sistema da formação com a base de dados do SIJ. Esta relação tem o objetivo de obter dados acerca dos tribunais e dos processos, bem como acerca dos *roles* e dos utilizadores existentes no SIJ.

Temos por último uma relação presente entre a aplicação Web da formação e a aplicação Web do SIJ. Esta é relativa à autenticação no sistema da formação com os utilizadores do SIJ. Assim, quando um utilizador deseja autenticar-se no sistema da formação, é redirecionado para a página de autenticação do SIJ e é aí que ele deve introduzir as suas credenciais. Assim o SIJ verifica se o

utilizador existe no sistema, e no caso de existir, devolve informação relativa ao mesmo para o sistema da formação. A partir desse momento o utilizador encontra-se autenticado no sistema da formação. Na **Figura 3.5** temos um diagrama de sequência onde se apresenta de uma forma detalhada este processo de autenticação.

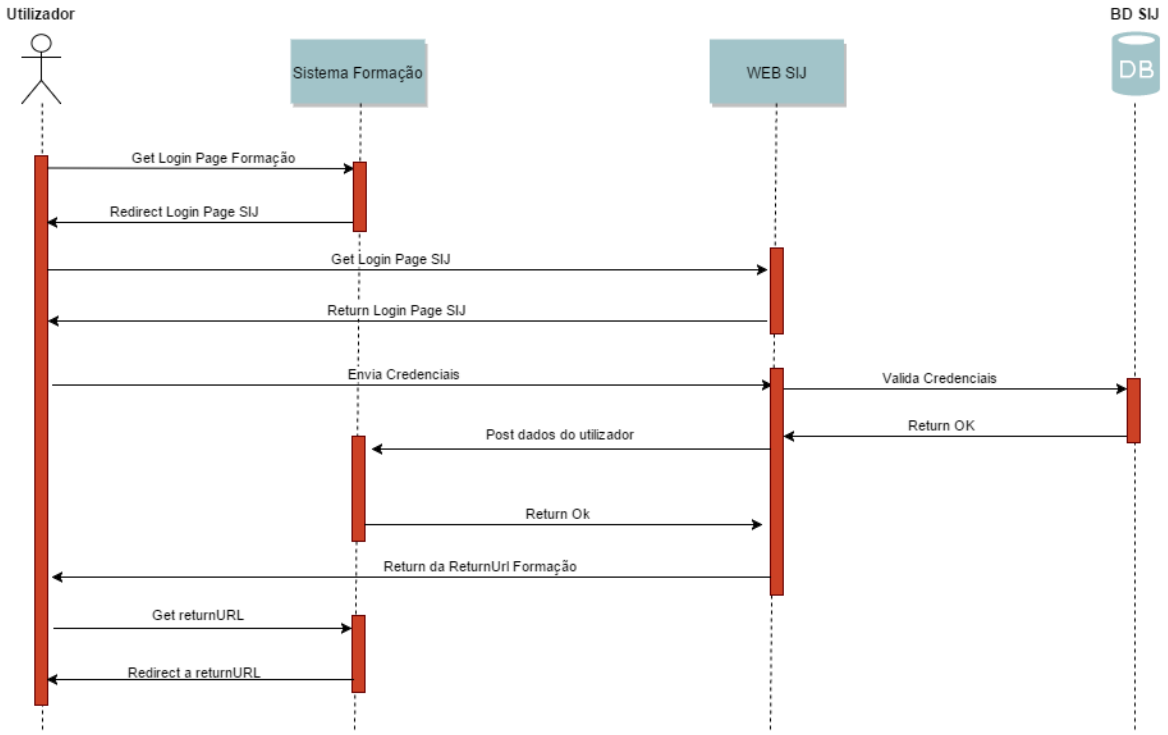

Figura 3.5 - Diagrama sequência para autenticação

Pode ver-se de uma forma detalhada o fluxo que existe para um utilizador se poder autenticar no sistema da formação. Portanto, quando o utilizador se deseja autenticar, faz um pedido ao sistema da formação, e o *site* da formação redireciona o utilizador para a página de login do SIJ com alguns parâmetros na *query string* (valor aleatoriamente criado no sistema da formação, valor que indica qual o sistema que se quer autenticar e a returnUrl). Aí o utilizador insere as suas credenciais e clica no botão para se autenticar o que faz com que as suas credenciais sejam enviadas para o SIJ. Já no SIJ, é feita uma consulta à base de dados para ver se o utilizador existe e se tem a respetiva *password*. No caso de isso acontecer, é enviado através de um *web service* para o sistema da formação por parte do SIJ, um JSON com o nome completo do utilizador, com o nome de utilizador, com uma lista de funções que este tem no SIJ e com uma variável que é o número aleatório anteriormente criado pela formação e enviado para o SIJ, de forma a manter os pedidos sincronizados. Havendo sucesso no pedido, os valores são guardados em sessão e o SIJ redireciona para a página do sistema da formação definida na returnUrl, que fora passada na *query string.*

Depois de esse redireccionamento ser feito, são obtidos os dados que foram enviados pelo SIJ que agora se encontram armazenados, e é criada uma sessão autenticada para o utilizador.

3.3 Modelo de Dados

3.3.1 Base de Dados

Relativamente ao modelo de dados definido para este sistema existem dois que são importantes de definir.

Um deles é o modelo de base de dados que é composto por 9 tabelas. A base de dados foi criada em Microsoft SQL Server e podemos dividi-la em duas partes. Uma parte é respetiva a dados estáticos que estão presentes desde o início e outra parte é respetiva a dados que vão sendo guardados consoante as atividades dos utilizadores.

É importante referir que este modelo de base de dados não é unicamente utilizado por este sistema, mas também pelo sistema que vai fazer a simulação das funções dos utilizadores e a monitorização das ações que os efetivamente estes vão fazer. Daí alguns campos podem não fazer sentido, pois a necessidade da sua existência veio do outro sistema.

O diagrama relativo à primeira parte pode ser visto na **Figura 3.6**. Deste diagrama fazem parte quatro tabelas que são descritas abaixo.

Descrevendo de uma forma rápida a utilidade destas tabelas, podemos dizer que a tabela “Fase” é onde estão armazenadas todas as fases processuais existentes no SIJ. A tabela “AcaoDisponivel” guarda todas as ações disponíveis, para o utilizador que está a criar os *templates* poder escolher. Esta e a tabela relativa às fases relacionam-se a partir da tabela “FaseAcaoDisponivel”. Assim é possível que uma ação esteja associada a uma ou mais fases, da mesma forma que uma fase pode estar associada a uma ou mais ações. Por último neste diagrama temos a tabela “Campo”. Esta tabela guarda os campos a preencher em cada ação. Podemos reparar que esta tabela tem uma relação para si mesma. O objetivo desta relação é possibilitar a existência de campos complexos, como o caso de objetos. Neste caso, um campo pode ser texto, número, data, booleano ou um objeto complexo que é composto por outros campos.

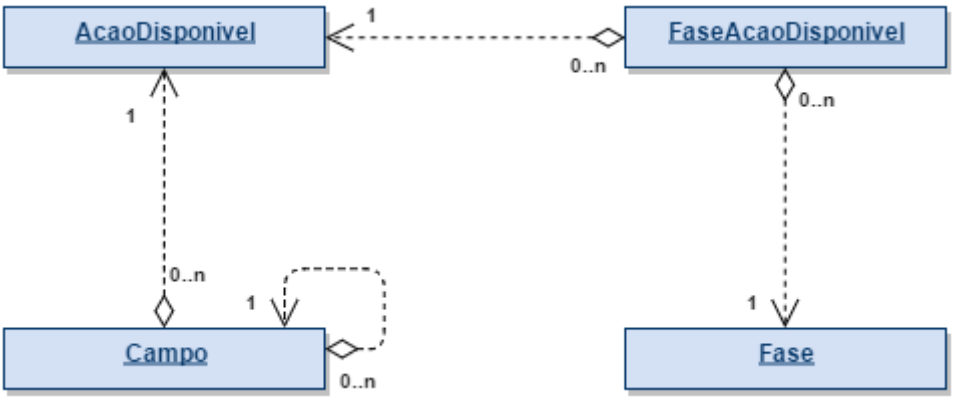

Figura 3.6 - Base de dados (parte relativa a campos e ações)

Na **Figura 3.7** temos um diagrama relativo à segunda parte da base de dados. Aqui podemos ver as tabelas necessárias para a criação e configuração de *templates*, bem como as tabelas necessárias para realizar a monitorização da execução dos mesmos. Neste diagrama temos a tabela "Template" onde são guardados os dados relativamente ao *template*. Nesta tabela existe um campo chamado "XML" onde é guardada uma estrutura em XML com todos os passos relativos a um determinado *template*. Cada um dos templates pode ter vários documentos associados, que são guardados na tabela "Documento", podendo assim serem descarregados durante a formação. Temos ainda a tabela "ConfiguracaoPerfisTemplate", onde é armazenada informação relativamente à configuração de *templates* para mais tarde poder ser arrancada uma formação com base nessa configuração. Aqui também temos um campo chamado "XML", onde são armazenados os passos, mas desta vez com uma configuração associada a cada um. Assim, é possível existirem várias configurações associadas a um *template*. Temos também a tabela "Instancia", que tem por objetivo guardar os dados relativos a cada uma das formações realizadas, com base na configuração definida na tabela "ConfiguracaoPerfisTemplate". Aqui é guardado o passo atual onde uma determinada formação se encontra, bem como a ordem desse passo no *template*. Nesta tabela é também guardada a data de início para armazenar a data em que começou a formação e a data de fim para o caso da formação já ter terminado, bem como a data do último passo realizado. Também existe nesta tabela um campo "XML" que serve para fazer um mapeamento de um determinado passo do *template* com o utilizador que o deve realizar. Por último temos a tabela "EstadoPorPasso", que serve para armazenar dados relativos à monitorização. Aqui, para cada um dos passos é guardada a data em que foi iniciado e, no caso de já ter terminado, a data em que terminou. É guardado também o utilizador que realizou o passo e se o passo foi bem executado ou não. E ainda é guardada

uma mensagem de texto relativa prestação do utilizador na execução do passo, para lhe poder apresentar.

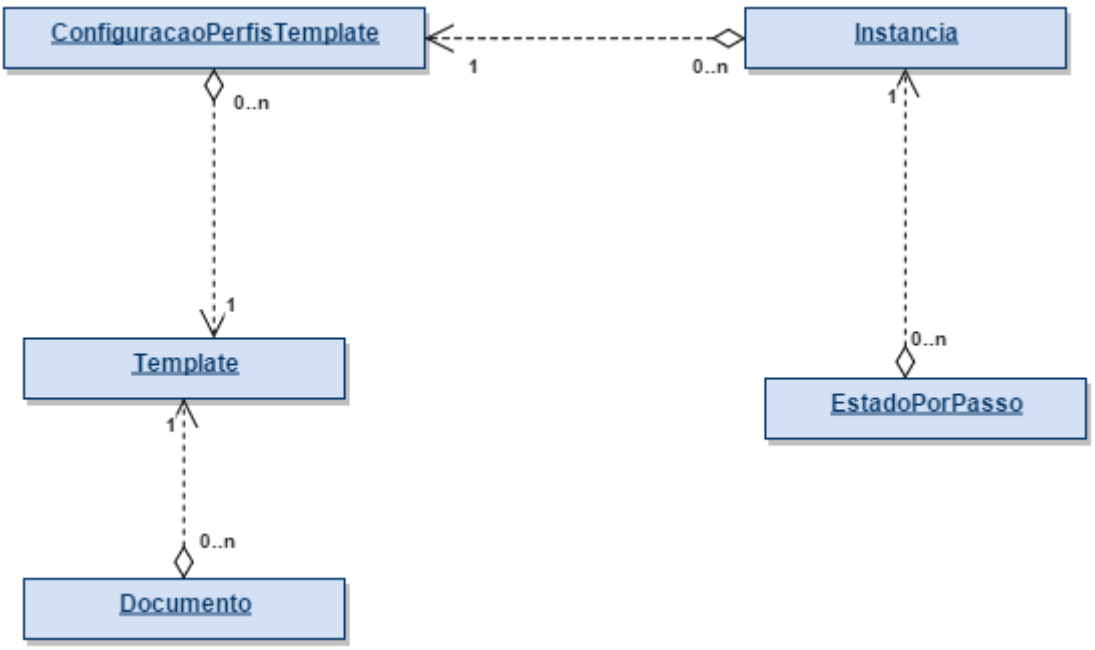

Figura 3.7 - Base de dados (parte relativa a atividades dos utilizadores)

3.3.2 XML Schema

Além do modelo de dados respetivo à base de dados, este projeto tem também um modelo de dados para documentos XML. Este surgiu devido à necessidade de termos um modelo de dados que fosse o mais flexível possível. Isto é, um modelo que possibilitasse uma alteração fácil, rápida e eficiente do modelo de dados devido à frequência com que é necessário alterar o modelo por causa da adição de novos passos para formações. Pois como todos os passos são diferentes, existe a necessidade de criar um novo objeto no *schema* para cada passo novo.

No caso de este modelo de dados ser adaptado a um modelo de base de dados iria obrigar à criação de muitas tabelas, trazendo uma grande dificuldade sempre que fosse necessário fazer uma alteração. No fundo, a cada passo novo introduzido no sistema seria necessária uma tabela. Ainda existe a grande vantagem de conseguir tornar os objetos existentes no *schema* em objetos C# com recurso a uma ferramenta chamada xsd.exe, conseguindo assim manipula-los com a utilização de *reflection*.

No sistema existem dois ficheiros relativos a *schema.* O primeiro, bastante mais extenso que o segundo, tem o objetivo de ter modeladas todas as tarefas possíveis, bem como todas as suas

propriedades para adicionar a um *template*. Para ser mais fácil de entender, são dados dois exemplos acerca da sua utilização. Começando primeiro com um exemplo mais simples, relativo à criação de um auto de denúncia, dando depois um exemplo mais complexo, relativo à explicação de um auto de denúncia.

Na **Figura 3.8** podemos ver o elemento "Acoes" a ser definido. Neste elemento irão ficar todos os passos de um dado *template*. Analisando o código, percebemos que este elemento tem um atributo que é o "tribunalId", onde deve ficar guardado o identificador do tribunal onde serão executas as tarefas definidas no *template*. O elemento "Acoes" ainda tem um tipo complexo que permite armazenar o número de vezes necessárias os tipos definidos ("AutoDenuncia", "ExplicacaoDoAutoDenuncia", "AutoDetencao", "ExplicacaoDoAutoDetencao").

```
<xs:element name="Acoes">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="AutoDenuncia">
        </xs:element>
        <xs:element ref="ExplicacaoDoAutoDenuncia">
        </xs:element>
        <xs:element ref="AutoDetencao">
        </xs:element>
        <xs:element ref="ExplicacaoDoAutoDetencao">
        </xs:element>
      </xs:choice>
      <xs:attribute name="tribunalId" type="guid"/>
    </xs:complexType>
  </xs:element>
```

Figura 3.8 - Elemento xsd Acoes

Neste documento XML *Schema* todos os elementos que são relativos a tarefas (os tipos complexos definidos em "Acoes") tem os atributos que estão listados na tabela abaixo.

Campo	Descrição
ordem	Ordem em que este passo vai ser executado.
descrição_passo	Uma descrição relativa ao passo.
id	Identificador único deste passo.
id_acao_bd	Identificador deste passo na base de dados.
nome_correto	Nome na forma correta para apresentar ao utilizador.
role	Função do utilizador que deve realizar este passo.
sistema	Indica se é o sistema ou o formando a realizar este passo.

Tabela 3 - Elementos relativos a tarefas

A **Figura 3.9** mostra-nos a definição do elemento "AutoDenuncia". Este é um tipo complexo e sendo um elemento relativo a uma tarefa tem todos os atributos listados na tabela acima. Além disso, é composto por mais cinco elementos simples: o número de documento, o autor, a data e hora, a descrição e a entidade originadora.

```
<xs:element name="AutoDenuncia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="NumeroDocumento" type="xs:string" />
        <xs:element name="Autor" type="xs:string" />
        <xs:element name="Data_Hora" type="xs:dateTime" />
        <xs:element name="Descricao" type="xs:string" />
        <xs:element name="EntidadeOriginadoraID" type="xs:positiveInteger"/>
      </xs:sequence>
      <xs:attribute name="ordem" type="xs:positiveInteger" />
      <xs:attribute name="descricao_passo" type="xs:string"/>
      <xs:attribute name="id" type="guid"/>
      <xs:attribute name="id_acao_bd" type="guid"/>
      <xs:attribute name="nome_correto" type="xs:string"/>
      <xs:attribute name="role" type="guid"/>
      <xs:attribute name="sistema" type="xs:string"/>
    </xs:complexType>
  </xs:element>
```

Figura 3.9 - Elemento xsd AutoDenuncia

Dando um exemplo um pouco mais complexo, vamos descrever o elemento "ExplicacaoDoAutoDenuncia" que é uma das ações possíveis de ser escolhidas. Este elemento é

também um elemento complexo e é relativo à tarefa que tem por objetivo a realização da explicação de um auto de denúncia. Tal como todas as tarefas, esta também tem os atributos que estão definidos na **Tabela 3**. Para além disso é composto por um elemento simples com o nome urgente, e por mais nove elementos complexos. São nestes elementos que vão ficar definidos os arguidos, os ofendidos, os meios de prova e os restantes valores necessários que caracterizam o auto de denúncia.

```
<xs:element name="ExplicacaoDoAutoDenuncia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Urgente" type="xs:boolean" />
        <xs:element ref="Arguido" maxOccurs="unbounded"/>
        <xs:element ref="Ofendido" maxOccurs="unbounded"/>
        <xs:element ref="Lesado" maxOccurs="unbounded"/>
        <xs:element ref="Testemunha" maxOccurs="unbounded"/>
        <xs:element ref="Interprete" maxOccurs="unbounded"/>
        <xs:element ref="Perito" maxOccurs="unbounded"/>
        <xs:element ref="Assistente" maxOccurs="unbounded"/>
        <xs:element ref="MOP" maxOccurs="unbounded" />
        <xs:element ref="MeiosProva" maxOccurs="unbounded" />
      </xs:sequence>
      <xs:attribute name="ordem" type="xs:positiveInteger" />
      <xs:attribute name="descricao_passo" type="xs:string"/>
      <xs:attribute name="id" type="guid"/>
      <xs:attribute name="id_acao_bd" type="guid"/>
      <xs:attribute name="nome_correto" type="xs:string"/>
      <xs:attribute name="utilizador" type="guid"/>
      <xs:attribute name="role" type="guid"/>
      <xs:attribute name="sistema" type="xs:string"/>
    </xs:complexType>
  </xs:element>
```

Figura 3.10 - Elemento xsd ExplicacaoAutoDenuncia

Utilizando o exemplo do arguido vemos que este também precisa de ser mais detalhado e não pode ser definido apenas por um elemento simples. Isto é, um arguido deve ter um nome e um número de BI definido. Sendo um arguido, tem de ter um advogado para o defender, um crime e outros aspetos importantes de definir. Assim o elemento “Arguido” é também um tipo complexo constituído por dois atributos, por dois elementos simples (nome e bi) e por quatro elementos complexos (“Advogado”,”Crime”,”MedidaGarantiaPatrimonial”, “MedidaDeCoacao”).

```
<xs:element name="Arguido">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Nome" type="xs:string" />
      <xs:element name="BI" type="xs:string" />
      <xs:element ref="Advogado" maxOccurs="unbounded"/>
      <xs:element ref="Crime" maxOccurs="unbounded"/>
      <xs:element ref="MedidaGarantiaPatrimonial" maxOccurs="unbounded"/>
      <xs:element ref="MedidaDeCoacao" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="identificador" type="guid"/>
    <xs:attribute name="descritivo" type="xs:string"/>
  </xs:complexType>
</xs:element>
```

Figura 3.11 - Elemento xsd Arguido

Utilizando agora o exemplo do advogado, que neste caso está relacionado com o arguido, este também necessita de ser definido com algum detalhe. Daí também ser um tipo complexo constituído por dois atributos e dois elementos simples relativos ao nome e ao número de cédula do advogado.

```
<xs:element name="Advogado">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Nome" type="xs:string" />
        <xs:element name="NumeroCedula" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="identificador" type="guid"/>
      <xs:attribute name="descritivo" type="xs:string"/>
    </xs:complexType>
  </xs:element>
```

Figura 3.12 - Elemento xsd Advogado

Analisando agora o segundo documento XML Schema podemos vê-lo completo abaixo.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="ConfigurationUserStep">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="UserStep">
        </xs:element>
      </xs:choice>
      <xs:attribute name="tribunalId" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="UserStep">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserID" type="xs:string" />
        <xs:element name="PassoId" type="xs:string"/>
        <xs:element name="RoleId" type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 3.13 - Documento xsd ConfigurationUserStep

O objetivo deste documento é a definição de um modelo para poder associar uma tarefa a um utilizador com um determinado *role*. Assim temos o elemento complexo "UserStep" que é composto pelo elemento "UserID", onde deve ser definido o identificador do utilizador que deve realizar a tarefa. É também composto pelo "passoId", onde deve ficar definido o identificador da tarefa. Por último temos o elemento "RoleId" que é onde se deve definir o *role* do utilizador que vai fazer a tarefa em questão. Outro elemento complexo é o "ConfigurationUserStep" que tem o atributo "tribunalId" definido. É aqui que deve ficar definido o identificador do tribunal onde serão realizadas as tarefas. Temos ainda uma referência para o elemento "UserStep". Estando esta referência dentro do elemento "*choice*" torna possível que o elemento "ConfigurationUserStep" tenha associado vários elementos "UserStep". O sentido de poder ter vários elementos associados deve-se ao facto de uma formação ter várias tarefas, com vários formandos, com *roles* diferentes. Assim, o elemento "ConfigurationUserStep" vai ter associado tantos elementos do tipo "UserStep" quanto o número de tarefas existentes no *template* em questão.

4 Desenvolvimento

Neste capítulo são abordados os detalhes mais importantes relativamente a implementação. E é ainda feita uma descrição do protótipo com recurso a imagens.

4.1 Implementação

A implementação deste projeto passou por fazer o levantamento das necessidades que foram pedidas. Depois de ter os requisitos bem definidos, tentou-se perceber qual seria o melhor caminho para desenvolver o projeto de modo a que cumprisse com todos esses requisitos da melhor forma possível. Então começou-se a idealizar um modelo de base de dados que fosse capaz de responder às necessidades já definidas. No entanto, cedo se percebeu que criar apenas um modelo de base de dados para sustentar toda a lógica iria tornar-se muito complexo visto que cada tarefa, bem como os campos que fossem do tipo complexo, iriam precisar de uma tabela. Para além disso seria difícil adicionar novas tarefas para a formação ou atualizar as já existentes, devido à necessidade de criar novas tabelas para o caso de se querer adicionar novas tarefas ou alterar as tabelas já existentes para o caso da atualização de tarefas. Era ainda necessário criar ou editar código para gerir essas tabelas adicionadas ou editadas, fazendo com que cada vez que se quisesse adicionar ou editar uma nova tarefa, era necessário alterar o *Schema* da base de dados e criar mais código. Assim, surgiu a solução de ter um modelo de dados baseado numa base de dados relacional e um modelo baseado em XML. Desta forma poderíamos criar um XML Schema, onde era definido o modelo para sustentar as tarefas e campos associados.

Utilizando assim esta solução, torna-se mais fácil adicionar ao sistema novas tarefas bem como a sua edição. No entanto o problema de ter que alterar o código cada vez que fosse necessário editar ou adicionar uma tarefa mantinha-se. Pois cada tarefa é diferente de outra tarefa, no que diz respeito aos campos que cada uma delas tem. Então, com recurso a uma ferramenta chamada xsd.exe é possível criar classes em C# a partir dos objetos e das suas relações existentes no XML Schema. Assim, surgiu a possibilidade de usar *reflection* para fazer a manipulação dessas classes e conseguindo dessa forma criar código suficientemente genérico que manipulasse todas as classes e respetivas propriedades que existem e que viriam a existir.

No entanto, definir o modelo de dados no XML Schema, não se revelava também suficiente. Havia informação que faltava, como por exemplo a ordem em que um determinado campo de uma determinada tarefa deveria aparecer no formulário. Assim, criou-se uma estrutura na base de

dados, onde se armazena a informação que não é possível ter no XML Schema. Essa informação é inserida na base de dados aquando a criação desta. Assim, a informação presente nesta estrutura é utilizada para listar os campos respetivos a uma determinada tarefa. Depois de listados e preenchidos chega a parte de armazenar os dados nos respetivos campos. É nesta situação que se utiliza *reflection* pela primeira vez no sistema. Na **Figura 4.1** está um excerto exemplificativo de código responsável pelo armazenamento dos dados nas respetivas propriedades dos objetos (criados com base no documento XML Schema). "NomeNormalizado" é uma variável do tipo *string* que tem o nome da classe (gerada a partir do documento XML Schema) que queremos instanciar. Na primeira linha é obtido o tipo de classe relativa à tarefa em questão. Na segunda linha, é criada uma instância de uma classe do tipo obtido na linha anterior. A seguir temos a operação "obj.GetType().GetProperties()" que vai devolver todas as propriedades dessa classe para que se seja possível iterar sobre elas com o *foreach*. A variável "valores", usada na primeira linha dentro do *foreach,* é uma lista de objetos compostos pelo nome do campo e o respetivo valor. Então a primeira linha dentro do *foreach* vai obter o objeto em que o nome da propriedade é igual ao nome da propriedade relativa à iteração atual. Depois disto, na segunda linha do *loop* é feita a atribuição do valor contido no objeto obtido na linha anterior à propriedade relativa à iteração atual.

```
Type type = Type.GetType(NomeNormalizado);

Object obj = Activator.CreateInstance(type);

foreach (var prop in obj.GetType().GetProperties())
{
 MyValues myValues=valores.FirstOrDefault(x=>x.NomeAtributo==prop.Name);
 prop.SetValue(obj, myValues.Valor, null);

}
```

Figura 4.1 - Exemplo de *reflection*

Depois de criada uma instância de um dado objeto e atribuídos os valores às suas propriedades, é feita uma serialização com recurso à ferramenta xsd.exe, que vai transformar o objeto que se encontra em C# num elemento XML, ficando este pronto a ser gravado na base de dados.

Por outras palavras e de uma forma mais resumida, a solução escolhida passa por ter um modelo, onde temos um documento XML Schema que define uma estrutura para as tarefas e os respetivos campos, e temos também uma base de dados relacional onde são guardadas informações extras

acerca das tarefas. Nessa mesma base de dados são guardados outros aspetos relacionados com as atividades dos utilizadores no sistema. Por exemplo, quando um utilizador cria um *template* de formação, são guardadas informações como o nome do *template*, a sua data de criação, o tipo de processo que vai ser tratado e ainda um documento XML (que cumpre as regras definidas no XML Schema) com os dados relativos às tarefas desse *template*. Para conseguir criar este XML utiliza-se a ferramenta xsd.exe que vai transformar a estrutura do XML Schema em classes C#, e objetos C# em elementos XML. Para gerir as propriedades das classes e atribuir-lhes valores é utilizado *reflection*.

No que diz respeito ao desenvolvimento da aplicação Web utilizou-se a *framework* ASP.NET MVC 5. Foi aqui que se criou efetivamente o modelo para criar a base de dados, pois foi utilizada a funcionalidade de C*ode First Migrations*. Esta funcionalidade permite a criação das tabelas e das suas relações na base de dados com recurso a código C#. Relativamente aos controladores, foi onde se desenvolveu toda a lógica necessária para este projeto, como por exemplo a criação, edição e remoção de *templates*, bem como a configuração de perfis e o arranque e monitorização das formações. Estas funcionalidades são apresentadas aos utilizadores através das *views*, que é o sítio onde foram criadas as páginas Web com recurso ao HTML, ao CSS e ao Javascript. De modo a termos um sistema bastante fluido, foram utilizadas com alguma regularidade chamadas assíncronas ao servidor, com recurso ao Javascript para que seja carregado apenas o que é necessário. Um exemplo disso, é quando um utilizador, na criação de um *template*, seleciona uma tarefa na *dropdownbox* e os campos relativos a essa tarefa são imediatamente apresentados ao utilizador sem que a página tenha sido carregada por inteiro.

Um outro aspeto que a que foi dada grande atenção foi a parte da monitorização de uma formação em tempo real. Aqui os utilizadores podem acompanhar o estado de cada tarefa numa dada formação. No entanto, surgiu a questão de os utilizadores acabarem a sua tarefas, mas isso não se refletir na interface da monitorização. Pois, normalmente quando uma página é carregada ela está atualizada tendo em consideração os dados presentes na base de dados existentes naquela altura. No entanto, se os dados forem alterados posteriormente, isso não é refletido na página até que esta carregue novamente. Assim, a informação apresentada sobre o estado de uma tarefa pode não estar correta numa determinada altura. Deste modo, surgiu a necessidade de atualizar a página Web quando houvesse alguma alteração das tarefas relativas à formação em questão. Para isso, poderia ser usada uma abordagem em que fossem feitas consultas à base de dados com regularidade para verificar se existia alguma alteração. No entanto, isso poderia trazer alguns problemas, sendo um deles a criação de uma carga desnecessária resultante de muitas consultas à

base de dados em poucos períodos de tempo. Para atenuar esse problema, poder-se-ia aumentar os intervalos de tempo para fazer menos consultas à base de dados. Mas aí voltaríamos ao problema inicial, ou seja, informação na página Web diferente da que existe na base de dados. Sendo assim, a solução foi utilizar SQLDependency e SignalR, descritos em 7.1. Com o SQLDependency torna-se possível criar um evento ao nível do C# de modo a que este seja acionado quando houver alguma alteração numa tarefa relativa ao *template* que se está a monitorizar. Desta forma é evitada a consulta consecutiva à base de dados. No entanto, é preciso refletir essas alterações na interface sendo aí que se utiliza Razor. Com Razor, é possível criar uma função em Javascript que esteja disponível para ser chamada pelo servidor. Assim, quando essa função for chamada, os dados atuais são obtidos e são atualizados os dados relativos às tarefas.

Importa referir que toda a solução está sobre o sistema de controlo de versões providenciado pelo *Team Foundation Server* (TSF).

4.2 Protótipo

Nesta secção será apresentado, com recurso a imagens, o sistema da formação de uma forma detalhada. No entanto, antes de fazer a apresentação do protótipo é importante perceber a logica de como funciona o sistema.

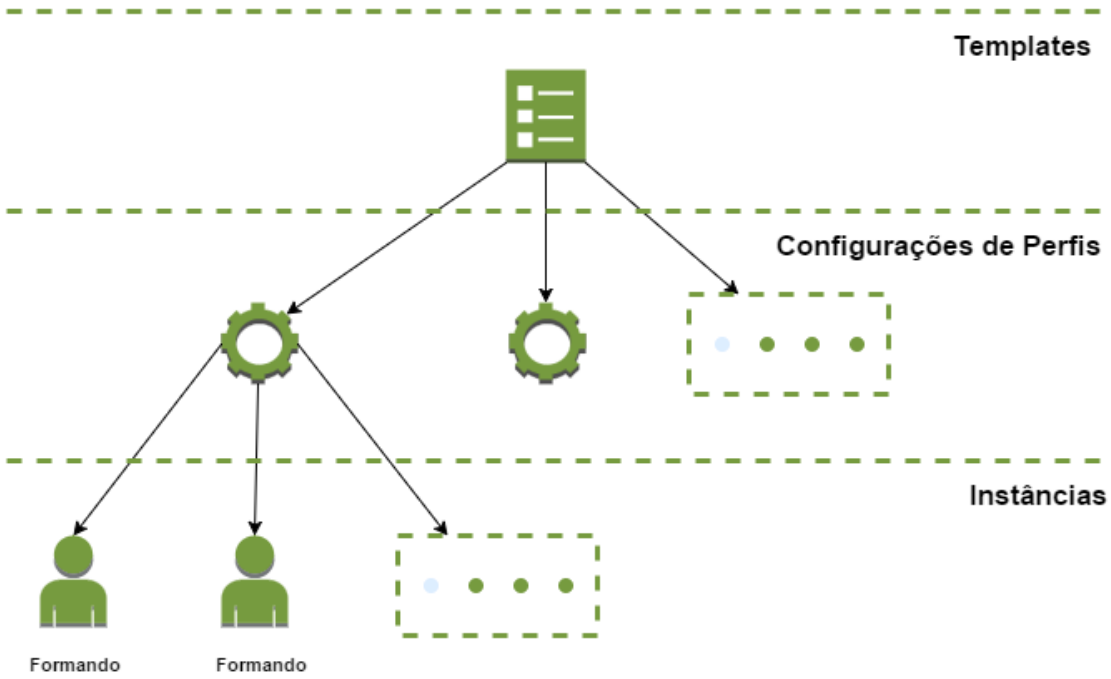

Figura 4.2 - Lógica do Sistema

Na **Figura 4.2,** podemos ver três camadas. Na camada primeira camada temos os templates, que são compostos por um conjunto de passos que devem ser feitos relativamente à tramitação de um processo. Na segunda camada, temos as configurações de perfis. Aqui, é definido qual o tipo de utilizador que deve realizar cada passo. Ainda para cada passo, deve ser definido se será um utilizador real a realizar esse passo, ou se este será simulado pelo sistema. Para cada *template*, podem ser realizadas inúmeras configurações de perfis. Na terceira camada temos as instâncias, que são relativas às formações iniciadas com base numa configuração de perfis antes realizada. Cada uma destas instâncias é relativa à formação de um formando. Assim, se for um formador a iniciar a instância, deverá para cada passo, selecionar especificamente que utilizador o deve realizar. Obviamente, esse utilizador selecionado deverá ter o *role* definido para esse passo na configuração de perfis. Se for um formando a iniciar a instância, este passará a realizar todos os passos que foram definidos para serem realizados pelo *role* dele, independentemente se foram configurados ou não para serem simulados pelo sistema. Já todos os passos definidos para serem realizados por outro *role* serão simulados pelo sistema. Para cada configuração de perfis, podem ser iniciadas quantas instâncias forem necessárias.

Na **Figura 4.3** podemos visualizar a primeira página. Aqui o utilizador pode escolher o que deseja fazer no sistema.

Figura 4.3 - Protótipo (Página inicial)

Podemos ver dentro do retângulo azul uma mensagem de boas vindas ao utilizador “Raf”, o que significa que este se encontra autenticado.

No centro da página podemos ver quatro opções. A opção de “Novo Template” que onde o utilizador poderá criar novos *templates* para formações, a opção “Listar Template” que como o próprio nome indica vai listar todos os *templates* que existem no sistema, a opção “Configuração de Perfis” que lista todos as configurações de perfis feitas para os *templates* e a opção de “Concretizações” onde são listadas todas as formações que já iniciaram.

4.2.1 Criar Template

Relativamente à criação de novos *templates*, é somente permitida para utilizadores que sejam administradores. Então, quando o utilizador clica na opção “Criar Template” na página inicial, é redirecionado para a página seguinte.

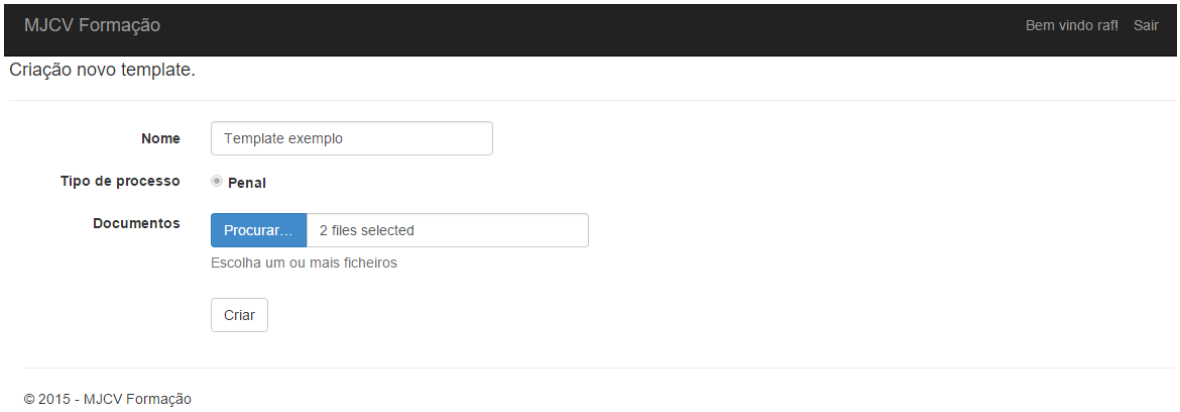

Figura 4.4 - Protótipo (criar template)

Nesta página o utilizador deve introduzir o nome que deseja dar ao *template*, deve definir o tipo de processo (ainda só é permitido escolher penal) e escolher documentos para fazer *upload*. Para terminar deve clicar em “Criar”. Será então criado um *template* na base de dados com o nome de “*Template* exemplo”, do tipo penal, com dois documentos. No entanto, ainda não foram definidas quaisquer tarefas para serem realizadas pelos formandos. Assim, depois de se clicar no botão “Criar”, o utilizador será redirecionado para a seguinte página.

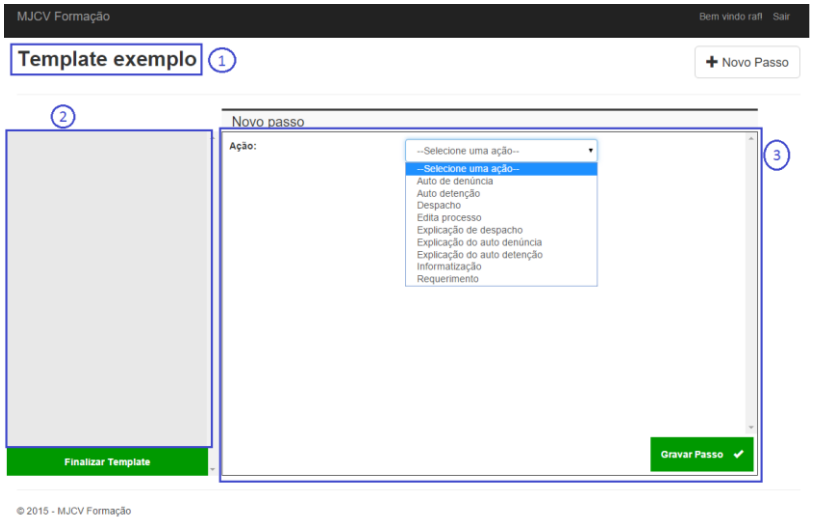

Figura 4.5 - Protótipo (criar tarefas inicio)

É nesta página que serão criadas as tarefas para os formandos realizarem. Podemos ver que ainda não existe qualquer tarefa definida. Relativamente à interface podemos ver em 1) o nome definido para este *template*, em 2) a lista de todos os passos já criados para este *template* e em 3) onde o utilizador escolhe a tarefa e preenche os respetivos campos associados. Vemos ainda um botão que permite gravar a tarefa depois de preenchidos os respetivos campos e um botão que permite finalizar este *template*. De modo a esclarecer melhor as capacidades desta funcionalidade, podemos ver abaixo dois exemplos de criação de duas tarefas. O primeiro exemplo é relativo à criação de um auto de denúncia, sendo esta uma tarefa relativamente simples. O segundo exemplo refere-se à explicação do auto de denúncia, sendo considerada uma tarefa mais complexa.

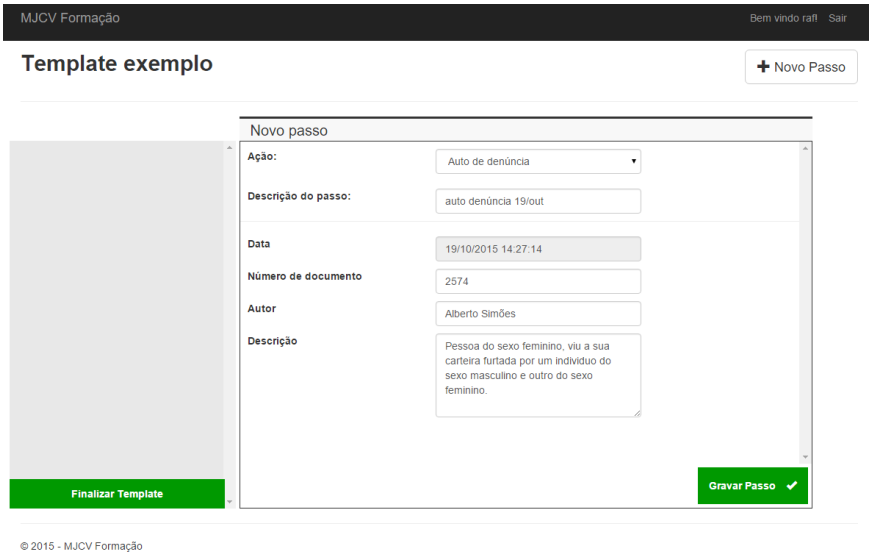

Figura 4.6 - Protótipo (criar auto denúncia)

Na **Figura 4.6** vemos o caso em que o utilizador escolheu a tarefa “Auto de denúncia”. Assim foram apresentados uma série de campos relativos a essa tarefa que o utilizador preencheu. Para efetivamente criar a tarefa, o utilizador deve clicar em “Gravar Passo”. A seguir irá ser listado na coluna das tarefas este auto de denúncia.

Na imagem abaixo é apresentado a criação da tarefa relativa à “Explicação do auto de denúncia”. Aqui podemos ver que a tarefa relativa à criação do “Auto de denúncia” já se encontra na coluna relativa às tarefas já definidas para este *template*.

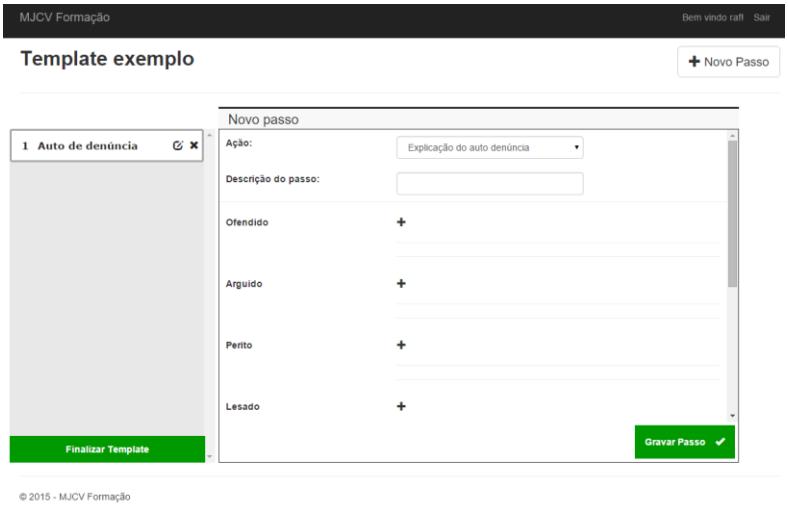

Figura 4.7 - Protótipo (criar explicação auto denúncia)

A tarefa relativa à explicação do auto de denúncia é mais complexa que a criação do auto de denúncia, pelo facto de ter campos associados que são campos complexos. Isto é, não são campos onde apenas se insere texto ou números, ou se escolhe um valor numa *dropdownbox* ou mesmo se seleciona uma *checkbox*, mas sim campos que contêm outros campos. O exemplo abordado vai ser a especificação de um ofendido. Assim, para se poder adicionar um ofendido, deve-se clicar no símbolo de “mais” relativo ao ofendido. Quando isso acontece, aparecem os campos necessários de preencher para o caso do ofendido, tal como vemos na **Figura 4.8**.

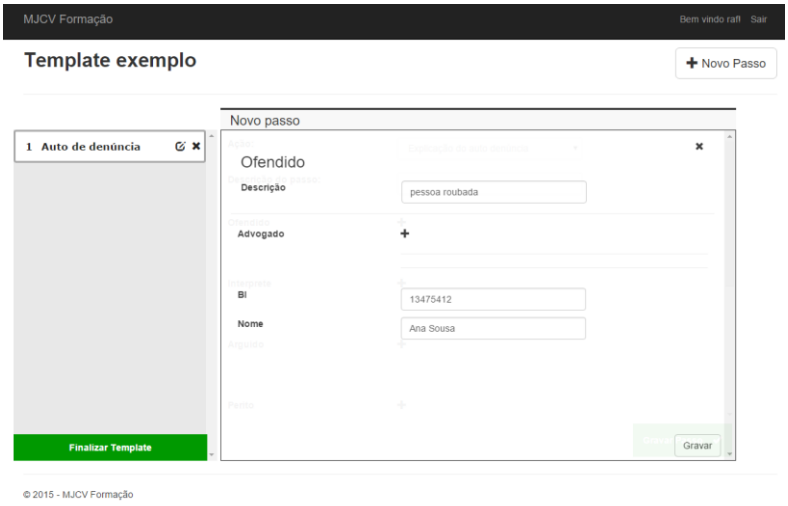

Figura 4.8 - Protótipo (adicionar ofendido)

Podemos ver que o ofendido tem um nome, tem um BI e também tem associado outro campo complexo que é o advogado. Assim, à semelhança do que foi feito para adicionar um ofendido, o utilizador deve clicar no símbolo de "mais" relativo ao advogado. Serão então apresentados os campos relativos ao mesmo, como podemos ver na **Figura 4.9**.

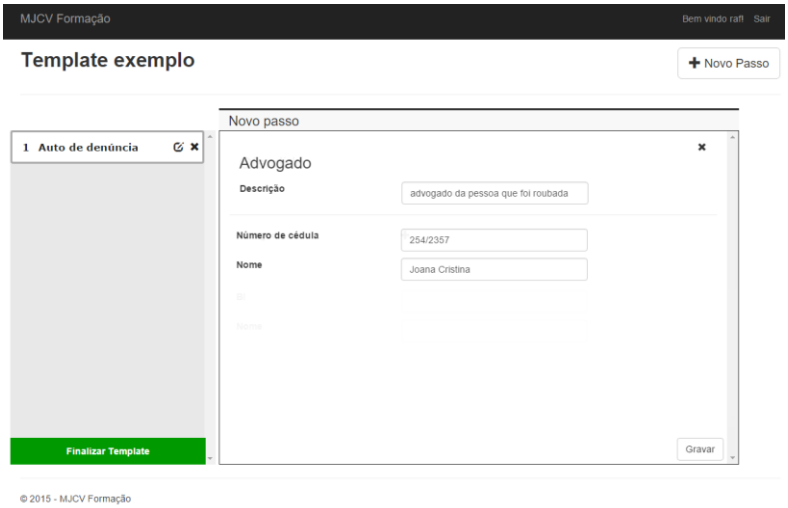

Figura 4.9 - Protótipo (adicionar advogado a ofendido)

Vemos então que o advogado é composto pelo nome e pelo número de cédula. Quando preenchidos estes campos, o utilizador deve clicar em "Gravar" sendo-lhe apresentado novamente os campos que compõem o ofendido, mas desta vez já com um advogado associado, como podemos ver na **Figura 4.10**.

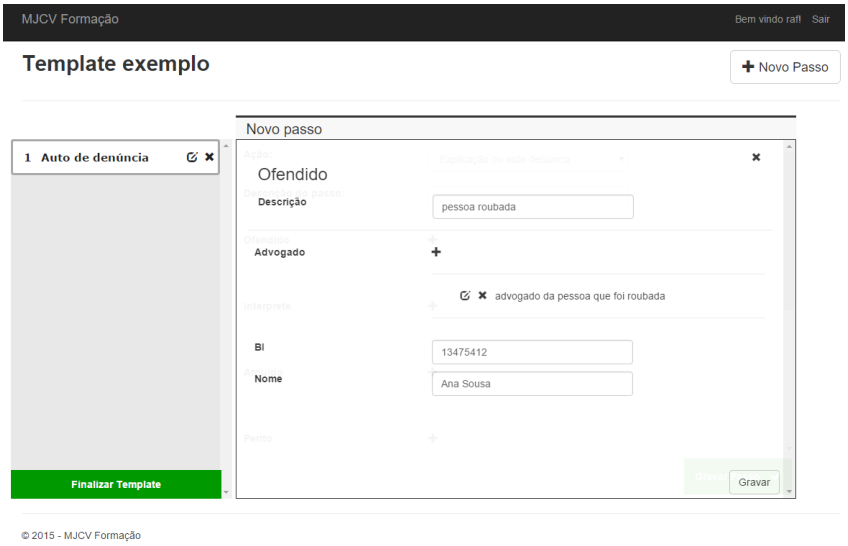

Figura 4.10 - Protótipo (ofendido criado)

É possível o utilizador adicionar mais do que um advogado. Pode ainda remover advogados já adicionados ou editá-los.

Assim, depois de preenchidos também os outros campos, o utilizador deve clicar em "Gravar" e serão então apresentados os campos relativos à explicação do auto de denúncia.

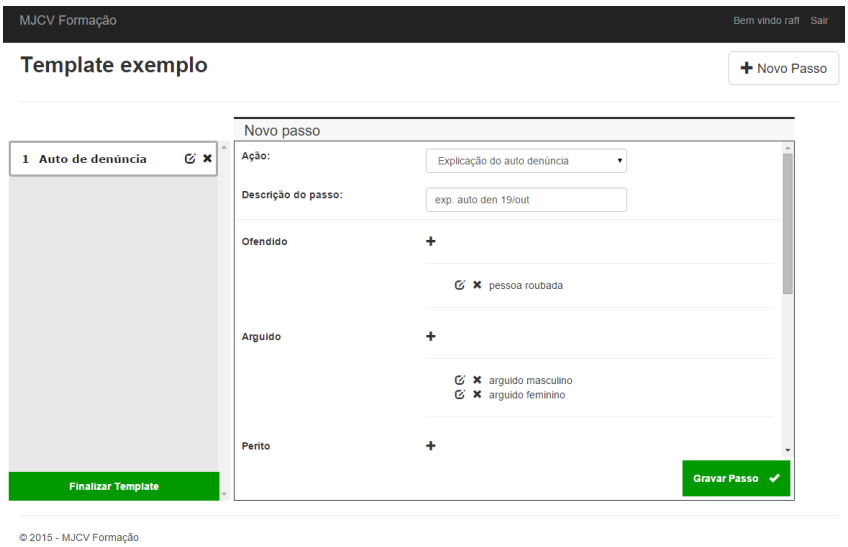

Figura 4.11 - Protótipo (explicação auto denúncia criada)

Na **Figura 4.11** podemos ver que a explicação de auto de denúncia já tem o ofendido associado. O mesmo foi feito para os arguidos, que neste caso foram dois. Para gravar esta tarefa o utilizador deve clicar em "Gravar Passo", ficando agora a coluna respetiva às tarefas com duas tarefas. É possível ainda, o utilizador alterar a ordem das tarefas arrastando-as para a posição pretendida com recurso a *drag-and-drop*. Podemos ver na **Figura 4.12** o utilizador a passar a tarefa relativa à

explicação do auto de denúncia para a primeira posição, de modo a que esta seja a primeira tarefa a ser realizada.

O utilizador pode adicionar quantas tarefas desejar. Para guardar este *template* o utilizador deve clicar em "Finalizar *Template*".

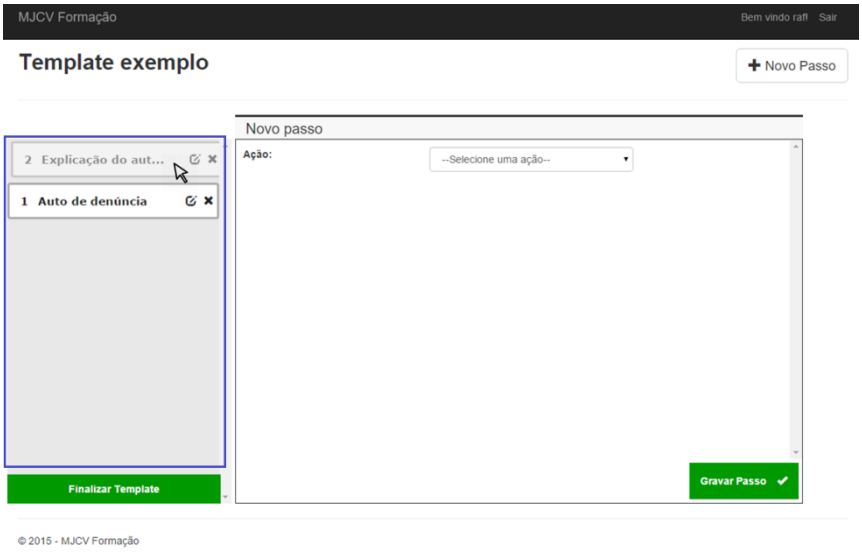

Figura 4.12 - Protótipo (funcionalidade *drag-&-drop*)

Para os passos seguintes será utilizado este *template*, mas com mais uma tarefa relativa a um requerimento.

4.2.2 Configurar perfis

Agora será apresentada a questão das configurações dos perfis dos utilizadores que devem realizar cada uma das tarefas. Esta funcionalidade também só pode ser realizada pelo administrador.

Na página inicial do sistema, o utilizador tem a opção de listar todos os *templates* já criados. Assim, se clicar nessa opção, será redirecionado para a página apresentada na **Figura 4.13**.

Aqui é apresentada uma lista com o nome, o número de tarefas, a data de criação e a data da última atualização, de todos os *templates* no sistema. Vemos que associado a cada *template* existem quatro ícones.

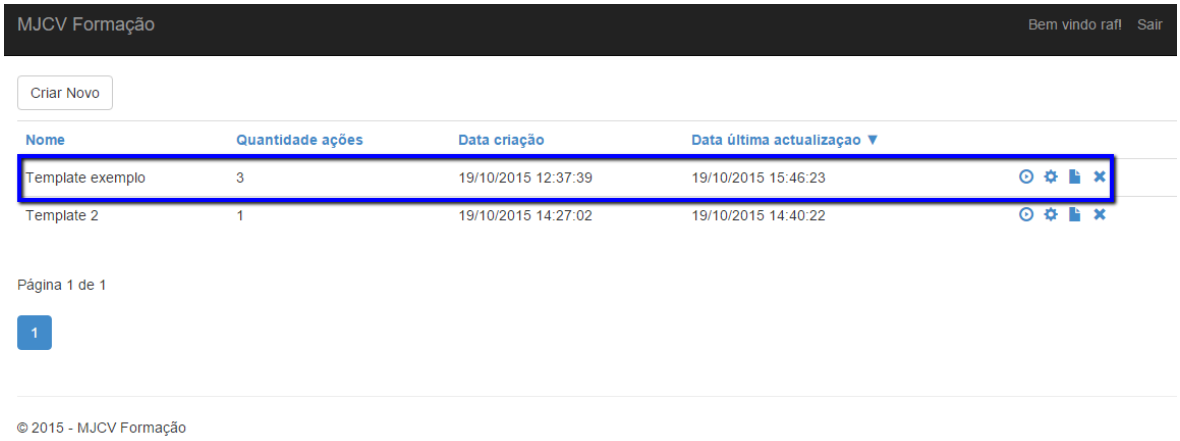

Figura 4.13 - Protótipo (lista de templates)

O primeiro ícone reencaminha o utilizador para uma página onde este pode configurar quais os *roles* dos utilizadores que devem fazer cada ação, bem como se é um formando a realizar a tarefa ou se é o sistema. O segundo ícone redireciona o utilizador para uma página onde poderá editar o *template* e as respetivas tarefas. O terceiro ícone reencaminha o utilizador para uma página onde este pode editar os documentos associados ao *template* em questão. O quarto ícone serve para o utilizador poder remover um determinado *template*.

Se o utilizador clicar no primeiro ícone, que é relativo à configuração dos perfis, será redirecionado para a seguinte página.

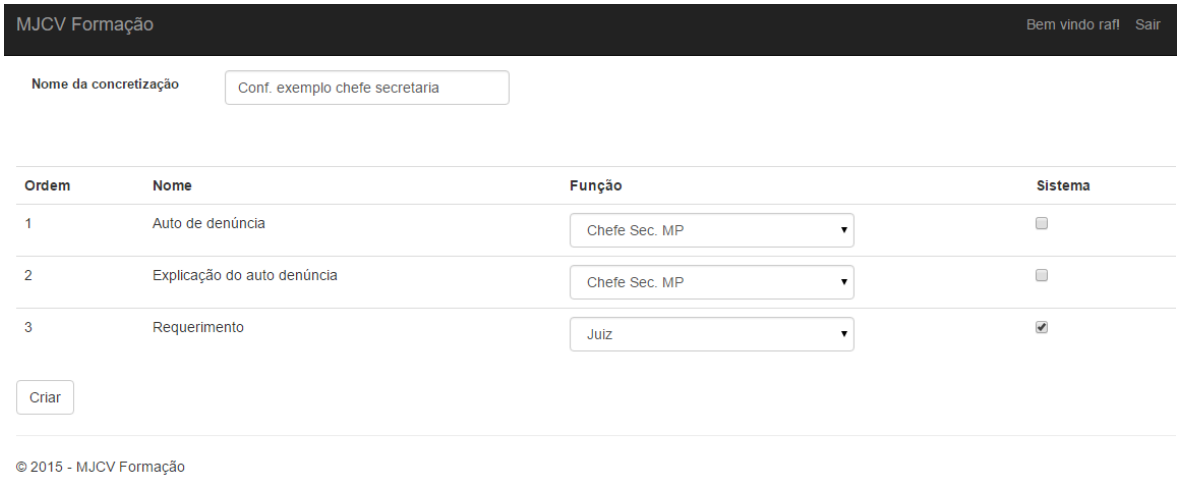

Figura 4.14 - Protótipo (configuração de perfis)

Podemos ver que nesta página temos uma caixa de texto para o utilizador dar um nome a esta concretização. Vemos também uma tabela com três linhas. Cada uma destas linhas corresponde a cada uma das tarefas que foram definidas no *template*. Em cada uma delas o utilizador tem uma *dropdownbox* com todos os *roles* existentes, de modo a que este possa escolher qual o *role* que

deve fazer uma determinada tarefa. Ainda para cada uma das tarefas, o utilizador deve selecionar a *checkbox* respetiva se essa tarefa vier a ser realizada pelo sistema. Assim, se for um formador a iniciar a formação todas configurações relativas a quem vai realizar a tarefa (se é o utilizador ou o sistema) mantêm-se. Já no caso de ser um formando a iniciar a formação, será verificado qual o seu *role*, e consoante o mesmo, serão alteradas as configurações de arranque para que apenas as tarefas que não estão associadas ao seu *role* sejam executadas pelo sistema, e as restantes executadas pelo próprio.

Ao clicar em "Criar" o utilizador será reencaminhado para uma página onde são listadas todas as configurações de perfis realizados.

4.2.3 Iniciar/Monitorizar

Chegamos ao ponto onde se pode iniciar uma determinada formação. Então, estando na página inicial, o utilizador deve clicar em "Configuração de Perfis" e será reencaminhado para a seguinte página.

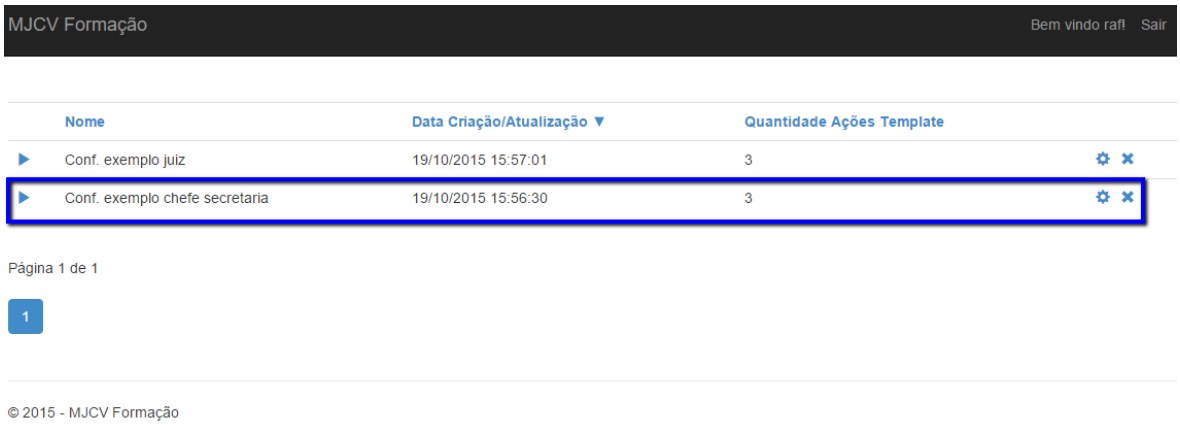

Figura 4.15 - Protótipo (lista de configuração de perfis)

Nesta página o utilizador pode ver uma lista com todas as configurações já definidas no sistema. Cada uma das linhas corresponde a uma configuração criada. Do lado direito podemos ver dois ícones. Ao clicar no primeiro, o utilizador é redirecionado para uma página onde este pode editar as configurações de perfis já definidas. O segundo ícone serve para o utilizador poder remover a configuração em questão. No início de cada linha podemos ver também um outro ícone. Este quando clicado faz com que seja iniciada a formação considerando as configurações definidas. Assim ao clicar nesse ícone o utilizador será redirecionado para a seguinte página.

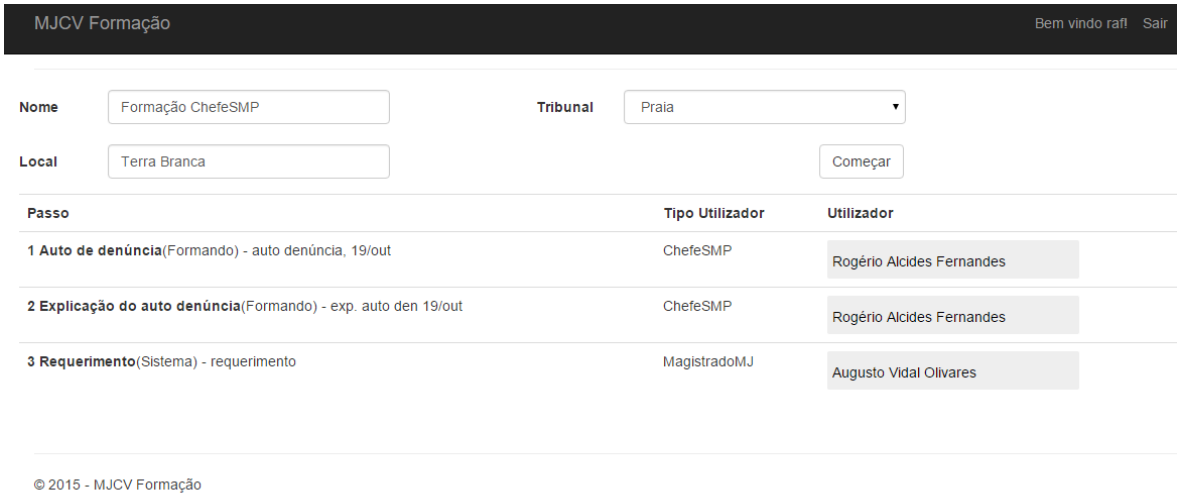

Figura 4.16 - Protótipo (iniciar formação)

Aqui, deve ser definido o tribunal onde estão os utilizadores que vão realizar as tarefas relativas à formação, bem como os próprios utilizadores. Para definir os utilizadores basta clicar sobre o respetivo campo, e será apresentada uma lista com todos os utilizadores daquele tribunal com o *role* em questão. Deve ser também definido um nome para esta formação e o local onde vai decorrer. Depois de tudo estar definido, o utilizador pode clicar em "Começar" para efetivamente ser iniciada a formação. Será então redirecionado para a página apresentada na **Figura 4.17**.

Estas funcionalidades estão disponíveis para serem realizadas tanto pelo administrador bem como pelo próprio formando. A diferença é que quando é o formando a iniciar a formação, todas as tarefas que estão configuradas para serem realizadas por um utilizador com um *role* igual ao dele, passam a ter de ser feitas por ele. E as restantes passam a ser feitas pelo sistema. Assim, as configurações feitas anteriormente no que diz respeito às configurações que indicam se é o sistema ou o formando a realizar a tarefa, ficam sem efeito.

Normalmente a primeira tarefa de uma formação deve corresponder à criação de um processo no SIJ. Assim, no caso de ser o formando a realizar essa tarefa e não o sistema de simulação, deve assim proceder à criação do processo e de seguida deve colocar o número de processo, que foi criado automaticamente pelo SIJ, na caixa de texto que se encontra na *modal.* Depois de introduzido, deve clicar em "Verificar se processo existe" para que possa haver uma validação deste. Esta validação consiste, em primeiro lugar, na verificação da existência do processo no SIJ. Depois verifica se foi introduzido pelo utilizador que foi configurado nas configurações de perfis para o introduzir, e por último verificar se a data da criação do processo é maior que a data relativa ao inicio da formação. Se o processo for validado com sucesso, o utilizador deve clicar em "Avançar"

para que a monitorização continue. No caso da primeira tarefa ser realizada pelo sistema, este passo, relativo à introdução do número de processo deixa de ser necessário.

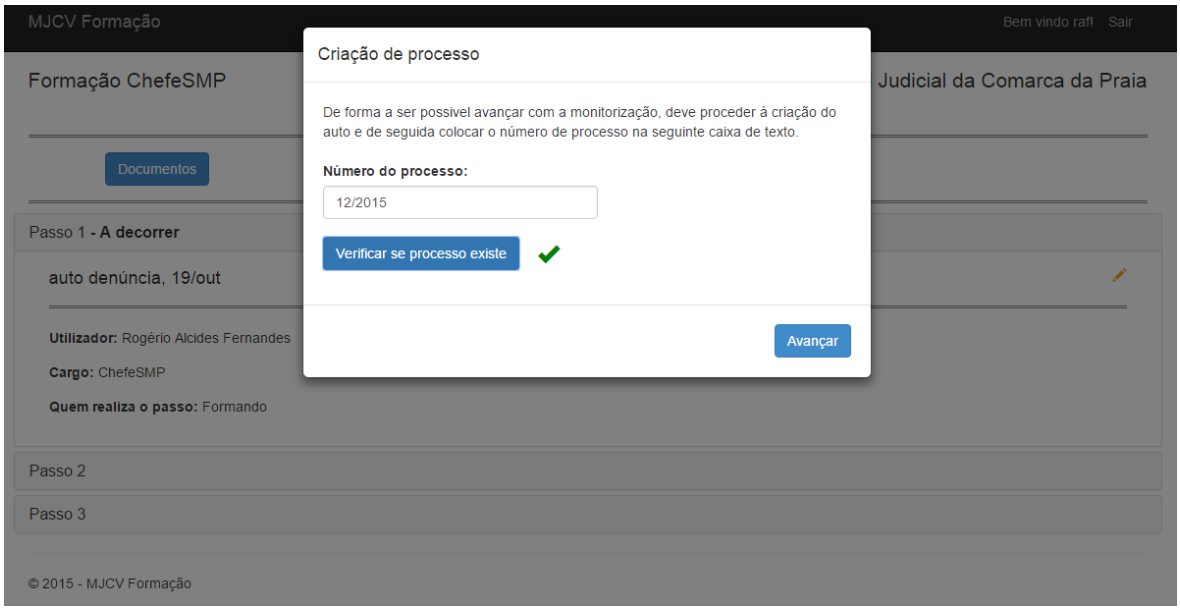

Figura 4.17 - Protótipo (Inserir processo em monitorização)

Na **Figura 4.18** podemos ver a monitorização de uma formação em tempo-real. Em 1) vemos o nome dado a esta formação e em 2) vemos o tribunal onde pertencem os formandos desta formação. Um pouco abaixo nessa página podemos ver três botões. O botão relativo a "Documentos" que serve para o utilizador poder descarregar todos os documentos que foram associados ao *template* relativo a esta formação aquando da sua criação. O botão "Terminar" que tem a funcionalidade de terminar esta formação. E o botão "Reiniciar" que serve para o utilizador poder reiniciar esta formação. Abaixo dos botões vemos um *accordion* com três painéis, em que cada um é respetivo a cada uma das tarefas definidas. O utilizador pode ver qual a tarefa que está a ser realizada nesse momento, pois no título do painel vemos o texto "A Decorrer", para que o utilizador consiga saber qual a tarefa que está a ser realizada num determinado momento, mesmo tendo os painéis todos fechados. Relativamente ao conteúdo dentro de cada um dos painéis é apresentada a descrição da tarefa, o utilizador e respetivo *role* que a deve realizar e se é o formando ou o sistema a realizar a tarefa. Existe ainda no canto superior direito um ícone que simboliza também o estado da tarefa. Isto é, se a tarefa já foi realizada com sucesso é apresentado um ícone verde, se a tarefa está a ser realizada, é apresentado um ícone amarelo e se a tarefa ainda se encontra em espera para ser realizada é apresentado um ícone azul.

Quando o formando termina a tarefa atual, a interface é atualizada de imediato fazendo com que informação apresentada nesta página esteja sempre atualizada. A título de exemplo, vamos imaginar o caso em que o formando termina a tarefa dois. Sem ser necessário o utilizador fazer *refresh* à página, ela é automaticamente atualizada, passando o texto "A Decorrer" para o título do painel relativo à terceira tarefa. Além disso, o ícone relativo ao painel dois passa a ser um ícone verde e o ícone relativo ao painel três passa a ser o ícone amarelo, que simboliza que essa tarefa está ser realizada nesse momento.

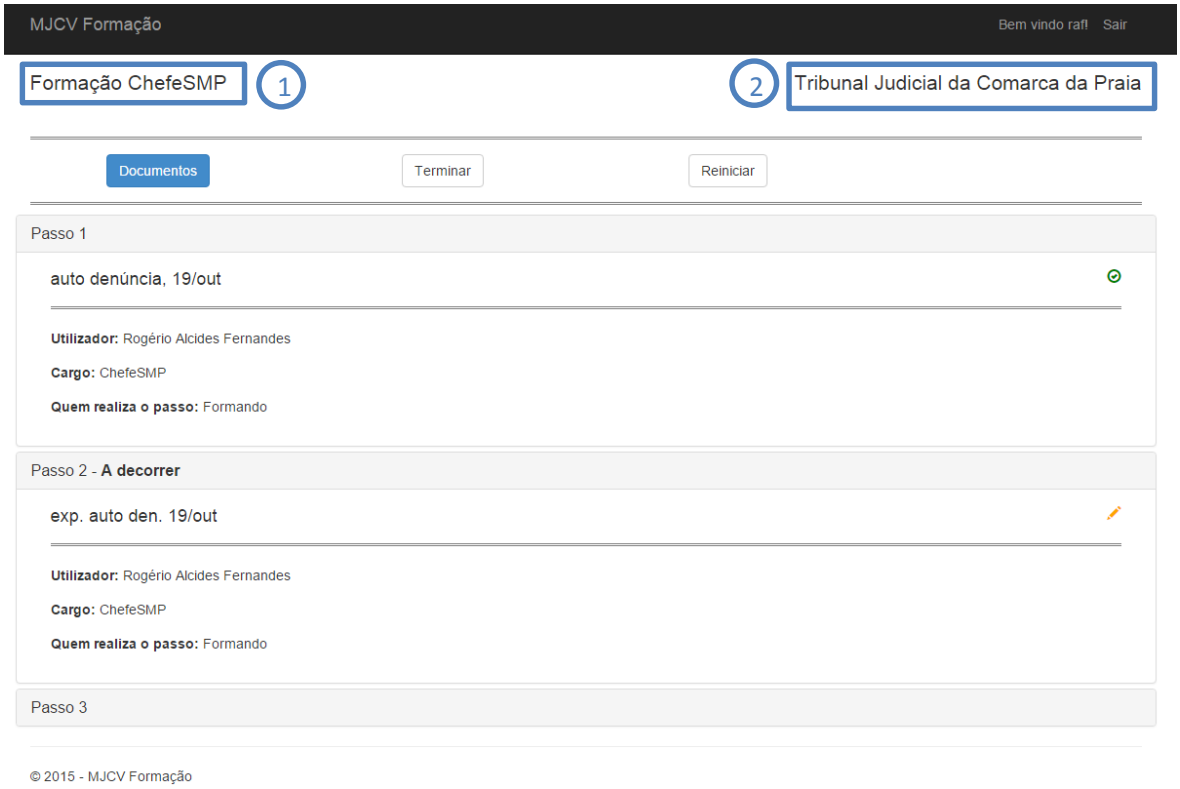

Figura 4.18 - Protótipo (Monitorização)

Existe ainda a possibilidade de serem listadas todas as formações já começadas. Assim, estando na página inicial (**Figura 4.3**), o utilizador pode clicar em "Concretizações" e será reencaminhado para a página que é apresentada na **Figura 4.19**. Aqui o utilizador pode filtrar os resultados que são listados de modo a aparecerem só as formações que já terminaram ou só as que estão a decorrer nesse instante. O utilizador pode ainda filtrar pelo local. Assim, à medida que é escrito o nome do local é feita uma filtragem apresentando apenas os resultados que cumprem o requisito imposto pelo filtro. No caso de a formação ainda estar a decorrer existe a hipótese do utilizador poder ser redirecionado para a página da formação (**Figura 4.18**).

Em todas as listagens presentes neste projeto, o utilizador tem a possibilidade de ordenar a lista por uma determinada coluna. Para isso basta clicar no título da coluna pela qual se quer ordenar.

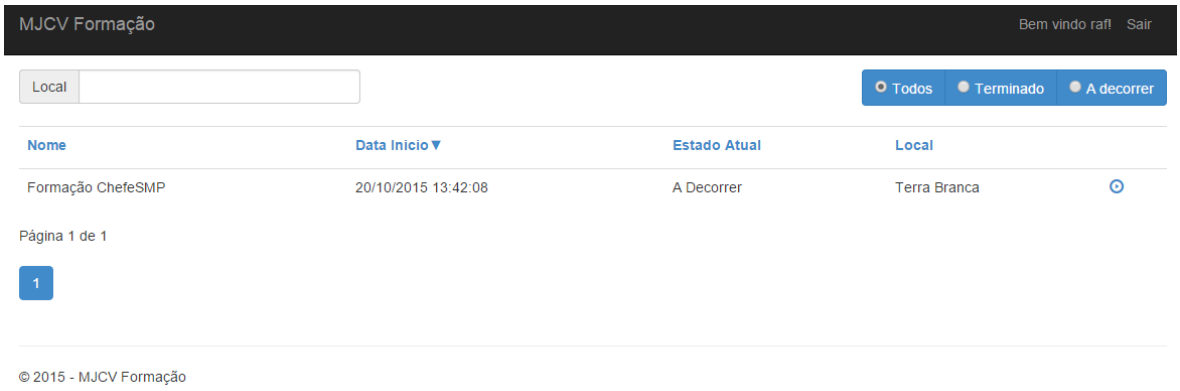

Figura 4.19 - Protótipo (Lista de formações iniciadas)

5 Conclusões

Para esta dissertação foi desenvolvido um sistema que permite a criação e gestão de formações para utilizadores do SIJ. Com este sistema, os responsáveis pela gestão das formações irão ver o seu trabalho facilitado relativamente à capacidade de criar previamente uma estrutura adequada a cada tipo de formando. Pode-se assim criar um *template* relativo a toda uma tramitação de um processo jurídico, podendo mais tarde configurar esse mesmo *template* de uma forma adequada aos tipos de formação que vão surgindo. Assim, a partir de um determinado *template* que seja constituído por tarefas relativas a vários tipos de *roles*, o responsável da gestão da formação pode configurar esse *template* para os formandos que deseja formar, selecionando as outras tarefas para serem simuladas pelo sistema. Isso pode ser feito para todos os tipos de utilizadores.

Este sistema permite formar vários formandos com recurso a uma sala e a um formador. Com este sistema, o formador pode ir acompanhando o estado das tarefas realizadas por cada formando de forma individual. Os próprios formandos podem consultar também as tarefas para essa formação, bem como o seu estado relativamente às tarefas da formação onde se encontra. Mais tarde, os formadores podem fazer consultas acerca do histórico relativo às formações, sabendo assim quais os formandos que realizaram determinadas tarefas, bem como o tempo que demoraram para as concretizar.

Para além das formações planeadas na sala, o formando pode, por sua iniciativa, realizar a formação em casa de modo a fortalecer todos os seus conhecimentos relativos ao sistema. Ou seja, o formando pode listar todas as formações que estão prontas a iniciar, independentemente de existirem ou não passos a serem simulados pelo sistema associados ao seu *role.*

De modo a responder às necessidades que levaram à implementação do sistema, criou-se um modelo de dados assente numa base de dados relacional e num documento XML Schema. No que diz respeito à criação da aplicação Web, esta foi desenvolvida com recurso à *framework* ASP.NET MVC 5, onde foi efetivamente criado o modelo para a base de dados relacional, e também desenvolvida toda a lógica para responder às necessidades dos requisitos anteriormente definidos. Aqui, ainda se criaram as páginas para apresentar aos utilizadores.

Visto este ser um sistema de apoio ao SIJ, foi criado um serviço REST para que o SIJ consiga enviar dados sobre um determinado utilizador, aquando o pedido de autenticação. No entanto, existem

outras situações onde acontece a comunicação com o SIJ, mas aí a ligação é realizada a partir da *Entity Framework.*

De acordo com o estudo realizado no estado de arte, foi assim possível desenvolver um sistema tendo em consideração a importância da formação em contexto organizacional, que possibilitasse não só ao formador, mas também aos formandos uma aprendizagem, que pode-se colmatar as diferentes dificuldades de cada individuo.

5.1 Trabalho Futuro

Ao longo do desenvolvimento deste projeto foram surgindo ideias relativamente a novas funcionalidades e a melhorias que se podiam fazer ao sistema.

Quando um administrador está a criar um *template*, este tem de saber rigorosamente quais as tarefas que são possíveis fazer a seguir a outras. Isto é, não tem sentido criar como primeira tarefa a realização de um requerimento, visto que este tem de estar associado a um processo. Assim, a primeira tarefa deve ser relativa à criação de um processo. Portanto, uma funcionalidade interessante seria desenvolver um motor de regras que garantisse que as tarefas disponíveis para adicionar a um *template* estivessem sempre dependentes de tarefas anteriormente selecionadas, permitindo assim que qualquer utilizador pudesse criar *templates* com sentido.

Outra questão semelhante que também era importante ser melhorada, é a questão da configuração de perfis. Esta funcionalidade permite a quem está a realizar a configuração de definir o *role* para fazer determinada tarefa. No entanto, nem todas as tarefas podem ser feitas por todos os *roles.* Assim seria importante haver também uma funcionalidade que dissesse quais os *roles* que podem fazer determinada tarefa, de modo a que não acontecesse uma situação de serem definidos *roles* que tivessem permissões para fazer a tarefa em questão.

Era ainda interessante este sistema ter a capacidade de criar guias para ajudar os utilizadores a realizarem as suas tarefas em tempo-real. Isto é, à medida que o utilizador fosse realizando os seus passos, existir sempre disponível uma ajuda que indicasse o que este deve fazer naquele momento.

Tendo em conta as sugestões dadas para trabalho futuro, seria possível desenvolver um sistema mais robusto, mais intuitivo, e que facilitasse as tarefas relativas aos gestores no que diz respeito à criação de tarefas, bem como a sua configuração.

6 Bibliografia

[1] J. Sousa Pinto e C. Teixeira, “Assisted On-Job Training,” em *E-Learning--Engineering, On-Job Training and Interactive Teaching*, INTECH, 2012, pp. 23-38.

[2] J. Brazeta, “Sistema de Informação da Justiça,” em *Testes de Software no Sistema de Informação da Justiça Cabo-Verdiana*, Aveiro, Universidade de Aveiro, 2014, pp. 6-11.

[3] whatishumanresource, “Training Process,” [Online]. Available: http://www.whatishumanresource.com/training-process. [Acedido em Setembro 2015].

[4] projectguru, “Difference between on the job and off the job training,” 6 Outubro 2012. [Online]. Available: http://www.projectguru.in/publications/difference-between-on-the-job-and-off-the-job-training/. [Acedido em Agosto 2015].

[5] whatishumanresource, “On-the-job Methods,” [Online]. Available: http://www.whatishumanresource.com/on-the-job-methods. [Acedido em Agosto 2015].

[6] businesscasestudies, “Business expansion through training and development,” [Online]. Available: http://businesscasestudies.co.uk/aldi/business-expansion-through-training-and-development/on-the-job-training.html#axzz3pCxBOYcf. [Acedido em Agosto 2015].

[7] yourarticlelibrary, “Methods of Training: On-the-job Training Method and Off-the-Job Methods,” [Online]. Available: http://www.yourarticlelibrary.com/human-resource-development/methods-of-training-on-the-job-training-method-and-off-the-job-methods/32369/. [Acedido em Agosto 2015].

[8] whatishumanresource, “Off-the-job Methods,” [Online]. Available: http://www.whatishumanresource.com/off-the-job-methods. [Acedido em Agosto 2015].

[9] A. F. d. Silva, Concepção de Formação em Contexto Real de Trabalho com Recurso a Ferramentas Avançadas, 2007.

[10] A. C. Almeida, O Tutor e a Formação Prática em Contexto Real de Trabalho, 2011.

[11] S. BRUM, “WHAT IMPACT DOES TRAINING HAVE ON EMPLOYEE COMMITMENT AND EMPLOYEE TURNOVER?,” 2007.

[12] smallbusiness, “The Effects of Lack of Employee Training,” [Online]. Available: http://smallbusiness.chron.com/effects-lack-employee-training-42687.html. [Acedido em Agosto 2015].

[13] A. Nassazi, “ HUMAN RESOURCE TRAINING AND DEVELOPMENT,” em *EFFECTS OF TRAINING ON EMPLOYEE PERFORMANCE. Evidence from Uganda*, 2013, pp. 21-22.

[14] getapp, “Get WalkMe vs Toonimo vs Whatfix Comparison,” [Online]. Available: https://www.getapp.com/it-management-software/a/walkme/compare/whatfix-vs-toonimo/. [Acedido em Setembro 2015].

[15] yourstory, “whatfix, a SaaS platform to create interactive guides raises INR 5.5 crores,” [Online]. Available: http://yourstory.com/2015/04/whatfix-funding-helion/. [Acedido em Setembro 2015].

[16] walkme, “This is WalkMe,” [Online]. Available: http://www.walkme.com/about-us/. [Acedido em Setembro 2015].

[17] W3Schools, “HTML Introduction,” [Online]. Available: http://www.w3schools.com/html/html_intro.asp. [Acedido em Agosto 2015].

[18] Mozilla Foundation, “Introduction to HTML,” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/Introduction. [Acedido em Agosto 2015].

[19] D. CAMERON, “JavaScript,” em *HTML5, JavaScript and jQuery*, Indianapolis, John Wiley & Sons, Inc., 2015, p. 95.

[20] Mozilla Foundation , “JavaScript basics,” [Online]. Available: https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/JavaScript_basics. [Acedido em Agosto 2015].

[21] W3Schools, “jQuery Introduction,” [Online]. Available: http://www.w3schools.com/jquery/jquery_intro.asp. [Acedido em Setembro 2015].

[22] D. Cameron, “jQuery Selection,” em *HTML5, JavaScript, and jQuery*, Indianapolis, John Wiley & Sons, Inc., 2015, p. 151.

[23] D. Cameron, “Introduction to CSS,” em *HTML5, JavaScript, and jQuery*, Indianapolis, John Wiley & Sons, Inc., 2015, pp. 27-30.

[24] Bootstrap, “Bootstrap,” [Online]. Available: http://getbootstrap.com/. [Acedido em Agosto 2015].

[25] J. Spurlock, “Bootstrap Scaffolding,” em *Bootstrap*, Sebastopol, O’Reilly Media, Inc, 2013, p. 1.

[26] microsoftvirtualacademy, “Introduction to ASP.NET MVC,” [Online]. Available: https://www.microsoftvirtualacademy.com/en-US/training-courses/introduction-to-asp-net-mvc-8322?l=gyInu8Zy_604984382. [Acedido em Setembro 2015].

[27] P. H. B. W. K. S. A. S. H. Jon Galloway, “THE RAZOR VIEW ENGINE,” em *Professional ASP.NET MVC 4*, Indianapolis, Wrox, 2012, pp. 57-58.

[28] I. Griffiths, “Introducing C#,” em *Programming C# 5.0*, O'Reilly Media, 2012, pp. 1-2.

[29] J. G. M. S. Christian Nagel, “.NET Architecture,” em *Professional C# 5.0 and .NET 4.5.1*, Indianapolis, Wrox, 2014, p. 3.

[30] Microsoft, “Reflection (C# and Visual Basic),” [Online]. Available: https://msdn.microsoft.com/en-us/library/ms173183.aspx. [Acedido em Agosto 2015].

[31] W3Schools, “Introduction to SQL,” [Online]. Available: http://www.w3schools.com/sql/sql_intro.asp. [Acedido em Agosto 2015].

[32] D. W. Keyvan Nayyeri, “Introduction to the Real-Time Web,” em *Pro ASP.NET SignalR*, Apress, 2014, pp. 1-9.

[33] Microsoft, “Detecting Changes with SqlDependency,” [Online]. Available: https://msdn.microsoft.com/en-us/library/62xk7953(v=vs.110).aspx. [Acedido em Agosto 2015].

[34] B. Lakshmiraghavan, “Building a Basic Web API,” em *Pratical ASP.NET Web Api*, Apress, 2013, p. 2.

[35] D. A. L. R. E. Q. Joe Fawcett, “What is XML?,” em *Beginning XML, 5th Edition*, Wrox, 2012.

[36] W3Schools, “Introduction to XML,” [Online]. Available: http://www.w3schools.com/xml/xml_whatis.asp. [Acedido em Agosto 2015].

[37] Microsoft, “How to: Use the XML Schema Definition Tool to Generate Classes and XML Schema Documents,” [Online]. Available: https://msdn.microsoft.com/en-us/library/5s2x1sy7(v=vs.110).aspx. [Acedido em Julho 2015].

[38] W3Schools, “CSS Introduction,” [Online]. Available: http://www.w3schools.com/css/css_intro.asp. [Acedido em Setembro 2015].

[39] walkme, “walkme,” [Online]. Available: http://www.walkme.com/. [Acedido em Setembro 2015].

7 Anexos

7.1 Anexo A - Tecnologias

Aqui são apresentadas todas as tecnologias e abordagens que foram utilizadas para o desenvolvimento do sistema Web apresentado anteriormente.

HTML

O HTML surgiu quando Tim Berners-Lee, no final dos anos 80 idealizava uma forma de partilhar documentos a partir da Internet. Assim a ideia era ter o conteúdo armazenado num determinado servidor de modo a que este estivesse disponível para ser transferido e visualizado num *browser*. Simplificando dessa forma o acesso a conteúdos e permitindo a disponibilização de conteúdos mais ricos a nível de imagem.

HTML (*HyperText Markup Language*) é a linguagem utilizada nos browser modernos e tem o objetivo de configurar o conteúdo das páginas Web com recurso a elementos HTML, a que se podem chamar *tags*. São estas *tags*, que definem a estrutura de uma dada página no que diz respeito à sua apresentação e ao conteúdo semântico [17].

Um exemplo simples é a *tag* relativa ao parágrafo (**Figura 7.1**) que é definida pela letra "p" envolvida pelo sinal de menor ("<") e pelo sinal de maior (">").

```
<p>Isto é um parágrafo</p>
```

Figura 7.1 - Elemento HTML

Depois de ser executado pelo *browser*, o que é apresentado ao utilizador é apenas o texto "Isto é um parágrafo". Portanto, as *tags* não são mais do que sinalizadores que indicam ao *browser* a forma como o conteúdo deve ser exibido. É importante ainda referir que é permitido que elementos tenham outros elementos dentro de si, possibilitando assim a formação de uma estrutura hierárquica [18].

Ainda relativamente às *tags*, é possível adicionar atributos. Esses atributos servem para acrescentar informação a uma respetiva *tag*. No **Figura 7.2** podemos ver que agora o parágrafo tem o atributo "id". Este atributo tem a finalidade de dar uma identificação única a este elemento dentro de todo o documento HTML.

```
<p id="paragrafo_exemplo">Isto é um parágrafo</p>
```

Figura 7.2 - Elemento HTML com atributo

Javascript/JQuery

Javascript é uma linguagem de programação suportada pelos *browsers* mais populares. Durante muito tempo Javascript foi visto como uma linguagem que apenas permitia implementar funcionalidades básicas, como por exemplo validar o que um utilizador inseriu num determinado campo. No entanto, nos últimos anos o Javascript cresceu bastante devido à capacidade que tem de melhorar o desempenho de um *site* Web. Hoje em dia é utilizado não só para fazer validações mas também para criar páginas com interfaces complexas [19].

O Javascript foi criado por Brendan Eich e pode ser definida como uma linguagem de programação dinâmica, que quando aplicada a um documento HTML possibilita a uma interação dinâmica por parte dos utilizadores [20].

No que diz respeito a aprendizagem da linguagem, o Javascript torna-se um pouco mais difícil de dominar relativamente ao HTML e CSS. Na **Figura 7.3** podemos ver um exemplo bastante simples onde é obtida a referência para o elemento HTML "h1", guardado numa variável chamada "myHeading" e de seguida é adicionado o texto "Olá javascript" à propriedade "innerHTML" da variável "myHeading", fazendo assim que o elemento "h1" apresente o texto "Olá javascript" [20].

```
var myHeading = document.querySelector('h1');
myHeading.innerHTML = 'Olá javascript';
```

Figura 7.3 - Exemplo JavaScript

Relativamente a JQuery podemos dizer que é uma biblioteca de código aberto escrito em Javascript com o objetivo de simplificar e manipulação de elementos HTML. JQuery é utilizado em muitas aplicações Web pois fornece uma API simples e intuitiva de aprender que permite realizar as tarefas mais comuns num desenvolvimento de um *site* Web. Esta biblioteca tem também o objetivo de resolver problemas que dizem respeito às questões de *cross-browser*, fazendo com que os programadores não tenham que se preocupar com as diferenças existentes em cada *browser* onde o site Web será apresentado [21] [22].

CSS

Enquanto o HTML é responsável por fornecer o conteúdo, o CSS (*Cascading Style Sheets*) é responsável por descrever como o conteúdo é apresentado. Sendo assim utilizado para fazer o *design* do *site* Web podendo ter em consideração os tamanhos de ecrãs existentes hoje em dia [23].

O CSS pode ser adicionado a uma página HTML de três diferentes maneiras (sendo a última a mais adequada) [23]:

- Adicionar o CSS dentro da *tag* HTML <style> ;
- Adicionar o CSS no próprio elemento HTML;
- Adicionar o CSS num ficheiro diferente e adicionar uma referência para esse ficheiro no HTML.

Na **Figura 7.4** podemos ver um exemplo de CSS. Esse exemplo está definido para o parágrafo com o id "paragrafo_exemplo" (**Figura 7.2**) que definimos quando se abordou o HTML.

```
#paragrafo_exemplo{
    color:blue;
    font-family:'Times New Roman';
}
```

Figura 7.4 - Exemplo CSS

Analisando o exemplo de CSS, podemos dizer que o texto relativo ao parágrafo vai ser azul com o tipo de letra "Times New Roman". E este estilo é aplicado ao elemento HTML com o id "paragrafo_exemplo", pois o símbolo de cardinal seleciona o elemento com esse id.

Bootstrap

Bootstrap é uma *framework* de *client-side* para o desenvolver *sites* Web responsivos. Esta *framework* torna o desenvolvimento mais rápido e fácil [24]. Este produto foi desenvolvido por Mark Otto e Jacob Thorton e foi lançado em 2011 quando os dois eram funcionários do *Twitter*. Tem vindo a crescer a nível de popularidade desde que foi lançado. Desde aí que também tem vindo a evoluir como produto, deixando de ser um projeto apenas orientado ao CSS para incluir também *plugins* Javascript, trazendo desta forma funcionalidades já criadas. Portanto, ao utilizar *Bootstrap* começamos já com bastantes estilos de CSS e algumas funcionalidades de Javascript criadas, tendo apenas de utilizar os elementos HTML certos com as respetivas classes [25].

ASP.NET MVC 5

O ASP.NET foi lançado em 2002 pela Microsoft. A ideia na altura era permitir aos programadores a criação de páginas Web seguindo os mesmos conceitos (*drag-and-drop*, eventos, etc) que já existiam para a criação de *Windows Forms,* sendo então este produto designado como ASP.NET *Web Forms.* No entanto, à medida que o tempo foi passando, pontos negativos foram aparecendo, nomeadamente a complexidade e a baixa qualidade do código que era gerado automaticamente e também o facto do ciclo de vida de uma página ser relativamente complexo.

Em 2009 surgiu como alternativa ao ASP.NET *Web Forms* o ASP.NET MVC. Este novo modelo assenta num padrão já conhecido de outras *frameworks* para a Web. Este tem uma grande vantagem que é a clara separação do código em três tipos. Assim, a finalidade do código de cada ficheiro define o tipo a que esse ficheiro vai pertencer. Ou seja, se estivermos perante o código responsável pela modelação, este ficheiro será guardado na seção dos *models*. Se o código for responsável pela apresentação de conteúdos ao utilizador, o respetivo ficheiro será guardado na seção de *views*. Por último, se estivermos perante um ficheiro com código relativo à lógica de negócio, este será guardado na seção relativa aos *controllers.*

Na **Figura 7.5**, temos um diagrama simples que representa um *request* de um utilizador num projeto ASP.NET MVC 5. Depois do utilizador fazer o pedido é feito um reencaminhamento para o *controller*, ou seja, para um determinado método de uma determinada classe que herde da classe System.Mvc.Controller. É neste método onde toda a lógica de negócio é tratada. Muitas vezes, a intenção de um determinado método é obter algum tipo de dados para apresentar no *browser* ao utilizador. Assim o método deve retornar um objeto para a *view.* Esse objeto que na imagem vemos como *Model,* não é mais que uma instância de uma classe já definida no *model*. É aqui que vão os dados para serem apresentados ao utilizador.

Uma *view* é um ficheiro composto por elementos HTML e por *Razor* (explicado mais a frente). Então, o seu papel é tratar dos dados que vem do *controller* e com a ajuda do *Razor*, combiná-los com o HTML para que os dados sejam apresentados ao utilizador de forma adequada. Outra forma de definir a *view* é classificando-a como um *template* com uma localização pré-definida no HTML para os dados que vem do *controller*.

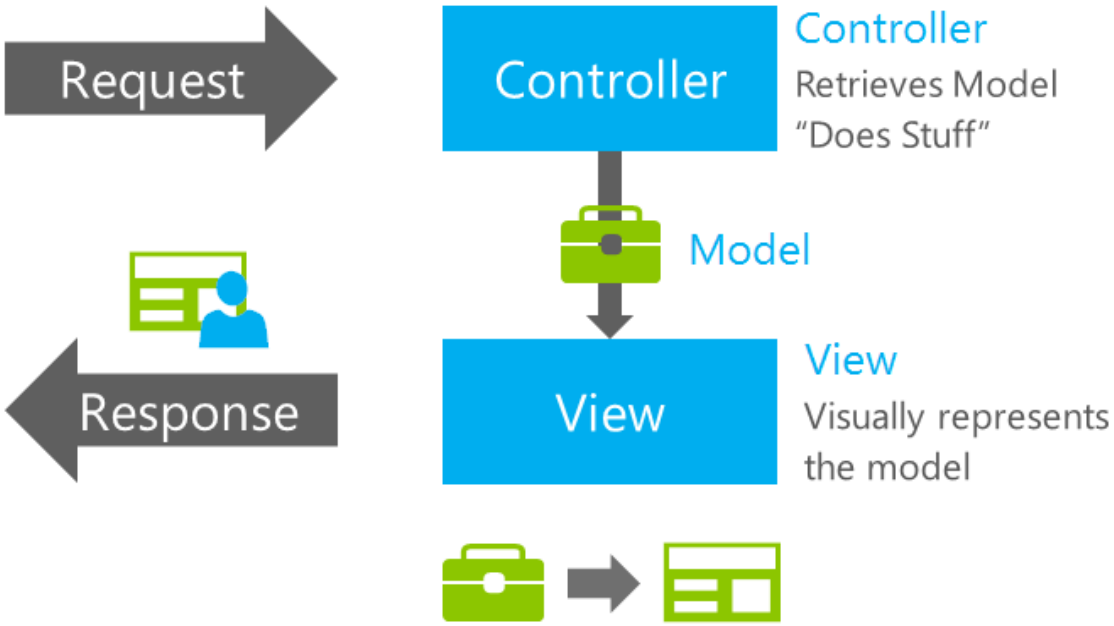

Figura 7.5 - Diagrama MVC **[26]**

Razor

Razor view engine foi introduzido com o ASP.NET MVC 3 e é definido como uma sintaxe para embutir no HTML das *views* [27]. Dessa forma, é possível combinar código e conteúdo que vem do *controller* do servidor de uma maneira bastante simples e intuitiva. Facilmente um programador com conhecimentos de C# ou VB consegue adaptar-se, pois Razor não é mais que código C# ou VB embutido em HTML.

```
<p>O mês de fevereiro no ano 2015 teve @DateTime.DaysInMonth(2015,2) dias.</p>
```

Figura 7.6 - HTML parágrafo

Uma das coisas essenciais em Razor é o sinal @. Este símbolo por norma indica que o código a seguir ao símbolo @ é código para correr do lado do servidor. De forma exemplificativa temos a **Figura 7.6** onde vemos código HTML relativo a um parágrafo. O que acontece é que no *browser* é escrito todo o texto que vemos da mesma forma, com a exceção de "@DateTime.DaysInMonth(2015,2)". Nesta situação o sinal @ transmite que "@DateTime.DaysInMonth(2015,2)" é uma instrução para ser executada do lado do servidor. Neste caso, esta instrução devolve o número de dias que tem o mês 2 do ano 2015, o que vai fazer com

que o texto dentro do parágrafo quando apresentado no *browser,* seja "O mês de fevereiro no ano 2015 teve 28 dias."

Um dos aspetos positivos de Razor é o facto de o símbolo @ ser interpretado com base no contexto. Isto é, o símbolo @ pode simbolizar que o texto seguinte é código para ser executado do lado do servido, mas também consegue ser suficientemente inteligente para conseguir interpretar o código que vemos na **Figura 7.7** corretamente e não assumir que ua.pt é uma instrução para correr do lado do servidor.

```
<a href="mailto:gandarez@ua.pt"></a>
```

Figura 7.7 - HTML Email

Sendo assim, as *views* são feitas com recurso a HTML e Razor, evitando assim os controlos complexos a que o ASP.NET *Web Forms* nos habituou, tornando o código das páginas para apresentar ao utilizador mais limpas e intuitivas.

C#

C# é hoje em dia a linguagem de programação principal das tecnologias Microsoft. Esta linguagem possibilita aos programadores desenvolver qualquer tipo de aplicação para o sistema operativo Windows. É uma linguagem desenvolvida tendo em conta os princípios de orientação a objetos que oferece um conjunto de componentes que permite um acesso eficiente a bases de dados [28] [29].

Reflection

Com *reflection* torna-se possível ler os metadados de um objeto em tempo real. Para isso é necessário utilizar o *namespace* System.Reflection. Assim é possível criar uma instância de um objeto, bem como obter um objeto já criado e conseguir fazer a sua manipulação em tempo real. Por exemplo, podemos ver quais as propriedades que constituem um objeto, ver o tipo de cada uma delas e obter o seu valor. Outro exemplo é a possibilidade de instanciar um objeto e chamar um dos seus métodos [30].

Como sabemos, o projeto desta dissertação tem como objetivo realizar um sistema que facilite a formação. Uma formação é planeada com base num conjunto de tarefas, em que cada uma tem as suas propriedades. Do ponto vista técnico, a ideia é a mesma. Cada tarefa é uma classe em C#. No entanto, a quantidade de possíveis tarefas implica um grande número de classes. Dessa forma não é viável estar a tratar as classes uma a uma. Até porque dessa forma cada vez que era necessário

adicionar uma nova tarefa à formação, era também necessário escrever mais código para conseguir lidar com essa nova classe. Assim a solução passou por utilizar *reflection*, que permite que todas as classes sejam manipuladas pelo mesmo código. Isto é, em tempo de execução o código consegue ver que tarefa é e fazer o tratamento adequado para cada uma delas. Facilmente se conclui que esta é a solução adequada, pois não era viável estar a fazer código à medida de cada uma das classes. Isso iria levar a um grande número de linhas de código. E ainda pior, quando comparamos ao facto de cada vez que queríamos adicionar uma tarefa nova teríamos de fazer mais código para lidar com a nova classe.

SQL

SQL (*Structured Query Language*) é um *standard* da ANSI (*American National Standards Institute*) e permite a manipulação de bases de dados. Esta linguagem permite fazer pesquisas sobre uma determinada base de dados de modo a obter os dados pretendidos, bem como inserir, atualizar ou remover registos. Permite ainda a criação de novas bases de dados, criar tabelas, procedimentos, *triggers*, *views* e funções [31].

No caso do projeto deste documento foi utilizado o Sistema de Gestão de Base de Dados (SGBD) SQL Server, que hoje em dia pertence à Microsoft.

SignalR

Damian Edwards e David Fowler eram dois funcionários da Microsoft que começaram um projeto *open-source* onde criaram uma biblioteca que facilitava a criação de aplicações Web com funcionalidades para comunicação em tempo-real. Dessa forma é possível um utilizador final navegar num determinado *site* e sempre que houver uma atualização, esta é imediatamente reproduzida na página. Um grande exemplo deste conceito é o Facebook. Quando alguém faz um comentário ou clica em "gosto" numa publicação feita por nós, recebemos essa atualização sem a necessidade de fazer *refresh* à página [32]. Outro exemplo são as plataformas para apostas desportivas durante os jogos. Nesses *sites* os prémios das apostas estão constantemente a mudar com base numa grande quantidade de variáveis.

SignalR tem três características principais [32]:

- Flexibilidade: Existem dois tipos de abordagens. Por um lado é oferecida uma forma mais simples e rápida de desenvolver aplicações com recurso a esta biblioteca, escondendo detalhes para facilitar o trabalho de quem está a desenvolver. Mas por outro lado são oferecidas ferramentas que dão mais flexibilidade e poder ao programador;

- Extensibilidade: Muitos componentes foram desenhados para serem substituídos por uma implementação mais adequada a uma determinada necessidade;
- Escalabilidade: providencia alguns mecanismos que os programados podem utilizar para escalar mais facilmente os seus sistemas

O que me levou a utilizar este conceito, foi para uma determinada parte do sistema relativa à monitorização de uma formação. Assim, os formandos e os formadores podem acompanhar em tempo real o estado de um determinado exercício, sem a necessidade de estarem constantemente a atualizar a página.

SQLDependency

SQLDependency permite a monitorização de uma base de dados a partir do código, possibilitando assim que sejam feitas notificações quando houver algum tipo de alteração. Evitando desta forma a criação de código com temporizadores que estejam regularmente a fazer consultas à base de dados para ver se existem algumas alterações [33]. Assim, é possível criar um evento no código (por exemplo C#) que é acionado quando alguma alteração acontecer, evitando as várias chamadas desnecessárias à base de dados. Desta forma é ainda garantido que no momento que algo é alterado, o evento é acionado, podendo assim responder em tempo real às situações e evitando que apenas depois do temporizador acabar se faça a verificação se houve alterações.

Web API

ASP.NET Web API é uma *framework* desenvolvida pela Microsoft para facilitar o desenvolvimento de serviços que sejam disponibilizados sobre HTTP, permitindo assim a possibilidade de ter serviços do tipo GET, POST, PUT e DELETE. Desta forma é uma boa opção para quem deseja criar aplicações REST em .NET. Embora seja possível também desenvolver serviços Web com *Windows Communication Foundation* (WCF), que é também uma *framework* desenvolvida pela Microsoft, a Web API começou a ser desenvolvida desde o início com o objetivo final de trabalhar sobre HTTP [34]. Enquanto a *framework* WCF foi inicialmente desenvolvida para trabalhar com SOAP e apenas mais tarde ajustada para trabalhar com REST.

Importa também referir que conceitos presentes no modelo ASP.NET MVC, como o caso de *routing*, *controllers*, validação, *model binding* encontram-se também na *framework* ASP.NET Web API.

XML

O *eXtensible Markup Language* (XML) encontra-se hoje em dia na maioria das aplicações de *software* desenvolvidas, o que implica que todos os programados devem conseguir compreende-lo

e usa-lo. O XML é também visto como uma solução para ter ficheiros legíveis por *software* e por humanos e foi desenvolvido para armazenar e transportar dados [35] [36].

XML foi lançado em 1996 e é uma recomendação do *World Wide Web Consortium* (W3C). Esta linguagem é um subconjunto de *Standard Generalized Markup Language* (SGML) e foi criada com o objetivo de se tornar mais simples que o SGML. XML não dita o formato de um ficheiro, apenas especifica regras o que lhe garante ter a flexibilidade que o SGML tem, mas sem a complexidade que lhe é conhecida [35] [36].

A título de exemplo, imaginemos que tínhamos de guardar o primeiro e último nome de uma série de utilizadores. Podíamos optar por escrever num ficheiro um nome em cada linha como podemos ver na **Figura 7.8**.

```
Ricardo Santos

Sandra Ribeiro

Vitor Costa

Ana Sobral
```

Figura 7.8 - Nomes Pessoas (texto livre)

Nós olhamos para o ficheiro e sabemos que são nomes apenas pelo conhecimento que temos relativamente aos nomes comuns que existem. Mas se olharmos para a **Figura 7.9**, já temos uma estrutura que nos dá mais detalhe. Uma pessoa que olhe para estas linhas consegue facilmente perceber o que significam. Relativamente a um sistema, facilmente pode ser programado tendo em conta o esquema definido [35].

```
<utilizadores>
	<utilizador primeiroNome="Ricardo" ultimoNome="Santos"/>
	<utilizador primeiroNome="Sandra" ultimoNome="Ribeiro"/>
	<utilizador primeiroNome="Vitor" ultimoNome="Costa"/>
	<utilizador primeiroNome="Ana" ultimoNome="Sobral"/>
</utilizadores>
```

Figura 7.9 - Nomes Pessoas (XML)

O XML que se encontra na **Figura 7.9** é composto por um elemento principal que é <utilizadores>, composto por sua vez por quatro elementos em que cada um é um utilizador. Cada um dos elementos respetivos a um utilizador tem dois atributos, que são o "primeiroNome" e o "ultimoNome".

Uma das desvantagens é o facto de os metadados fazerem com que o tamanho do ficheiro aumenta muito. Comparando o exemplo da **Figura 7.8** com o exemplo da **Figura 7.9**, vemos que o exemplo sem metadados tem 49 caracteres e o outro tem 247 caracteres. O que significa que o ficheiro com metadados tem aproximadamente cinco vezes mais caracteres.

Para terminar, podemos ver três cenários onde o XML é utilizado [35]:

- Ficheiro de configuração: a maioria dos ficheiros de configuração hoje em dia utiliza XML. Um exemplo é a web.config, ficheiro de configuração para projetos Web em .NET.
- Web *services*: embora exista também a opção de utilizar JSON, o XML é também utilizado para passar dados entre aplicações independentemente das tecnologias em que estas foram desenvolvidas. Os w*eb services* do tipo SOAP são descritos com recurso ao XML, dando a origem a um ficheiro chamado WSDL.
- Representação de imagens – imagens vetoriais podem ser representadas com XML, trazendo a vantagem de poderem ser manipulas mais facilmente.

XML Schema

O XML Schema é uma alternativa ao DTD para descrever a estrutura e o conteúdo de documentos XML de uma forma mais detalhada, permitindo uma validação mais precisa. É criado utilizando sintaxe XML enquanto as DTD's utilizam uma sintaxe diferente. Proporciona a capacidade de fazer validações com base em tipos já definidos, como por exemplo inteiros, booleanos ou texto, e também com base em tipos definidos pelo utilizador. Permite ainda a fácil criação de modelos de conteúdo e a sua reutilização. XML Schema divide os tipos de dados em duas categorias, que são os tipos simples e os tipos complexos. Sendo os tipos simples os elementos que definem o tipo de dados e o nome, enquanto os tipos complexos podem ter atributos e outros elementos associados a si. Na **Figura 7.10** podemos ver dois tipos simples, que são o elemento com o nome "MPDesiste" que define um tipo booleano, e o elemento com o nome "MotivoDesistencia" que define um tipo *string*. Podemos ver ainda um tipo complexo composto por dois atributos (identificador e descritivo) e pelos dois elementos "MPDesiste" e "MotivoDesistencia".

```
<xs:element name="Desistencia">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="MPDesiste" type="xs:boolean" />
        <xs:element name="MotivoDesistencia" type="xs:string" />
      </xs:sequence>
      <xs:attribute name="identificador" type="guid"/>
      <xs:attribute name="descritivo" type="xs:string"/>
    </xs:complexType>
</xs:element>
```

Figura 7.10 - Exemplo XSD

XSD.exe

XSD.exe é uma ferramenta que permite transformar os objetos definidos num dado documento XML Schema, em classes prontas a serem utilizadas na linguagem de programação C# e permite também serializar objetos para XML [37].

Para utilizar esta ferramenta deve-se abrir a linha de comandos, navegar ate ao local onde se encontra o ficheiro XML Schema e executar o comando "XSD <Nome do ficheiro> /d", como exemplificado na **Figura 7.11** sublinhado a azul. No caso de tudo correr bem é gerado um ficheiro relativo a C#, como vemos também na **Figura 7.11** sublinhado a verde.

```
C:\ieeta_projects\MJCVFormacao\MJCVFormacao\MJCVFormacaoWebApp\OtherDocuments> xsd ConfiguracaoUtilizadorPasso.xsd /d
Microsoft (R) Xml Schemas/DataTypes support utility
[Microsoft (R) .NET Framework, Version 4.0.30319.33440]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'C:\ieeta_projects\MJCVFormacao\MJCVFormacao\MJCVFormacaoWebApp\OtherDocuments\ConfiguracaoUtilizadorPasso.cs'.
```

Figura 7.11 - Utilização xsd.exe

Então, se o ficheiro relativo ao XML Schema for constituído pelo conteúdo apresentado na **Figura 7.12**, depois de executado o comando teríamos um ficheiro com duas classes de C#, que podemos ver na **Figura 7.13**.

```
<?xml version="1.0" encoding="UTF-8" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="ConfigurationUserStep">
    <xs:complexType>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element ref="UserStep">
        </xs:element>
      </xs:choice>
      <xs:attribute name="tribunalId" type="xs:string"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="UserStep">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="UserID" type="xs:string" />
        <xs:element name="PassoId" type="xs:string"/>
        <xs:element name="RoleId" type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Figura 7.12 - XSD Exemplo

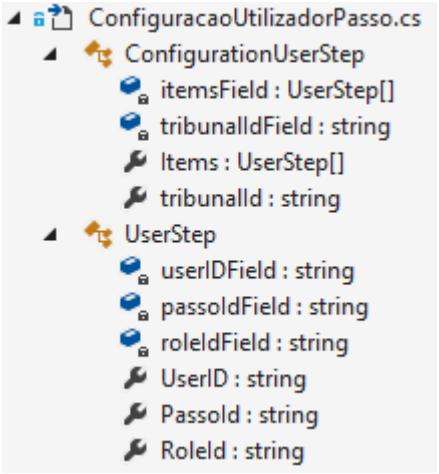

Figura 7.13 - Classe ConfigurationUtilizadorPasso

Se olharmos para o conteúdo do documento XML Schema, podemos ver que temos um elemento chamado “UserStep” que contem três elementos do tipo *string*. Agora se olharmos para a estrutura do ficheiro gerado podemos ver que existe uma classe chamada “UserStep” com três propriedades do tipo *string* e com os mesmos nomes que tinham os elementos no documento XML Schema. Por último, vemos ainda um elemento no documento XML Schema chamado “ConfigurationUserStep”,

que contem um atributo chamado “tribunalId” do tipo *string* e que tem um elemento complexo com referência para o elemento “UserStep”, permitindo assim que sejam incluídas varias instâncias desse elemento. Olhando agora para a estrutura do ficheiro gerado automaticamente vemos que existe uma classe chamada ConfigurationUserStep, com uma propriedade do tipo *string* chamada “tribunalId” e com uma outra propriedade capaz de guardar varias instâncias da classe “UserStep”.

Selectivity

Selectivity é uma biblioteca Javascript desenvolvida com recurso a JQuery e a Zepto.js. Esta biblioteca é compatível com todos os *browsers* modernos e permite a criação de *dropdownbox's* com funcionalidades acrescidas. Foi escolha para este projeto devido à facilidade de implementar um filtro dinâmico associado aos valores apresentados no elemento. Desta forma, depois de a *dropdownbox* ser expandida, será disponibilizado um campo de texto que permite ao utilizador ir escrevendo de acordo com o item que este deseja selecionar. Assim, à medida que o utilizador vai escrevendo, os dados apresentados no elemento vão sendo filtrados a partir do texto introduzido.

Importa ainda referir que esta libraria utiliza o *bootstrap*, isto é, por omissão os elementos tomam estilos já definidos do *bootstrap*.

Podemos ver na **Figura 7.14** e na **Figura 7.15** um exemplo do uso desta biblioteca. Na **Figura 7.14** é apresentada uma lista com todas as opções associadas à *dropdownbox.* Na **Figura 7.15** podemos ver que o utilizador escreveu “li” na caixa de texto o que vai atuar como filtro sobre todas as opções e dessa forma devolver apenas as opções onde existe correspondência.

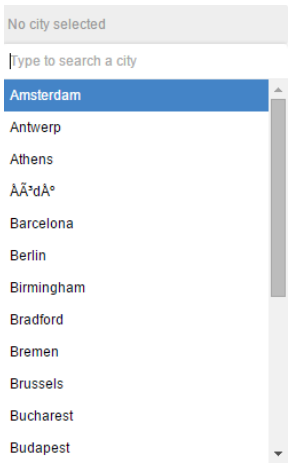

Figura 7.14 - Selectivity (sem filtro)

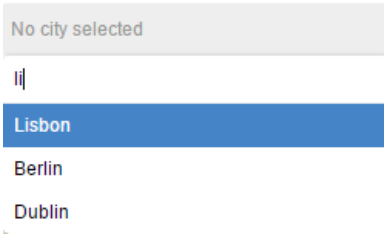

Figura 7.15 - Selectivity (com filtro)

Bootstrap-DatetimePicker

Este componente é um componente que permite aos utilizadores escolher a data e hora para colocarem num determinado campo. Assim, este componente providencia as funcionalidades para o utilizador final selecionar uma data e providencia também um aspeto gráfico apelativo, que é baseado em *bootstrap*.

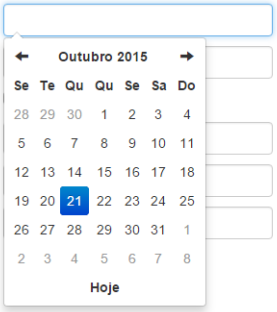

Figura 7.16 - DateTimePicker

7.2 Anexo B – Tabelas na Base de dados

Neste anexo é apresentado o modelo de base de dados utilizado e são também descritas detalhadamente as tabelas existentes no modelo de base de dados.

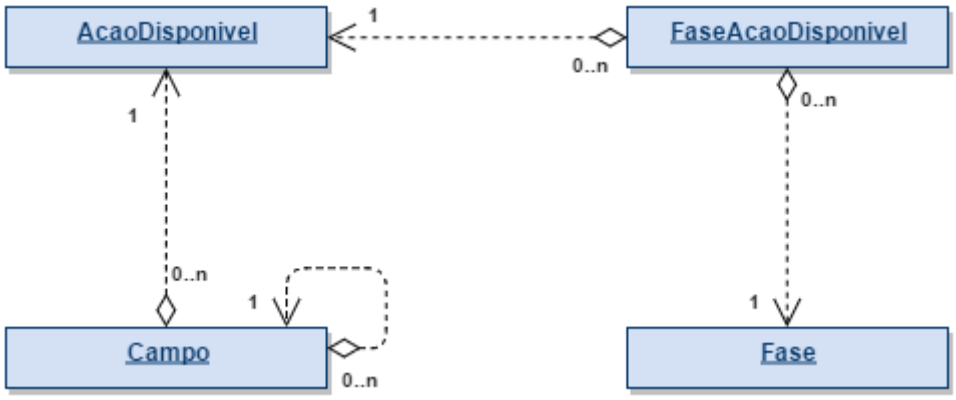

Figura 7.17 - Base de dados (parte relativa a campos e ações)

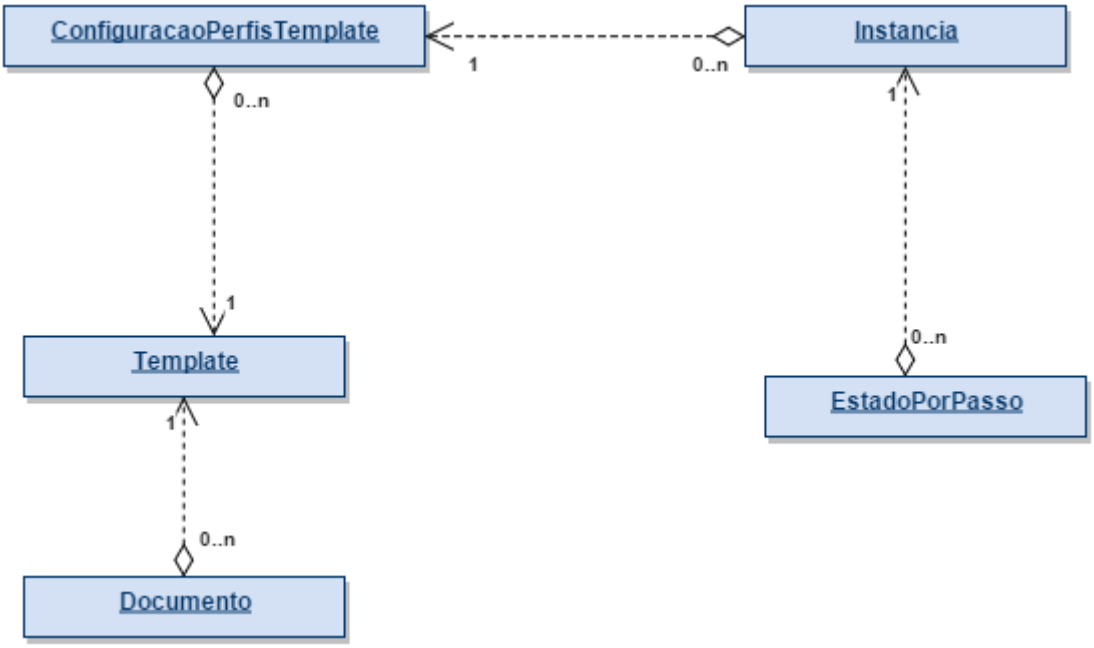

Figura 7.18 - Base de dados (parte relativa a atividades dos utilizadores)

A tabela “Fase” existe com o propósito de armazenar todas as fases que um processo no sistema de informação da justiça pode ter.

Fase		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
Nome	nvarchar(max)	Campo onde fica guardado o nome da fase.

Tabela 4 - Tabela Fase

A tabela "AcaoDisponivel" guarda todas as ações disponíveis que o utilizar vai poder escolher para cada passo no *template*.

AcaoDisponivel		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
Nome	nvarchar(max)	Campo onde fica guardado o nome de cada ação.
NomeNormalizado	nvarchar(max)	Campo onde fica guardado o nome de forma normalizada. Isto é, o nome apenas com caracteres entre A e Z e entre 0 e 9.

Tabela 5 - Tabela AcaoDisponivel

A tabela "FaseAcaoDisponivel" guarda as relações existentes entre a tabela "Fase" e a tabela "FaseAcaoDisponivel". Esta tabela existe devido à relação entre as outras duas tabelas ser de "muitos-para-muitos", o que obriga a que exista uma tabela intermédia. Aqui é armazenado se uma determinada ação, numa determinada fase está associada ao ministério público ou ao juízo.

FaseAcaoDisponivel		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
MP	nvarchar(max)	Campo que indica se esta ação pertence ao ministério pública ou ao juízo.
AcoesDisponiveis_ID	uniqueidentifier	Campo que relaciona esta tabela com a tabela AcaoDisponivel.
Fase_ID	uniqueidentifier	Campo que relaciona esta tabela com a tabela Fase.

Tabela 6 - Tabela FaseAcaoDisponivel

A tabela "Campo" guarda os campos que são necessários para cada Acão. Aqui são guardados os campos bem como o seu tipo de dados, a ordem em que vão aparecer listados e uma *regular expression* para validar o valor inserido.

Campo		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
Ordem	int	Campo que vai definir a ordem em que este campo irá aparecer na interface.
Nome	nvarchar(max)	Campo onde fica guardado o nome de cada ação.
NomeNormalizado	nvarchar(max)	Campo onde fica guardado o nome de forma normalizada. Assim, este campo apenas contem valores com caracteres entre A e Z e entre 0 e 9.
TipoVariavel	int	Campo onde é guardado um valor respetivo a uma enumeração que classifica o tipo de variável correspondente a este campo

Regex	nvarchar(max)	Campo onde é guardado uma *regular expression* de modo a saber que valores são admissíveis para este campo.
Obrigatorio	bit	Indica se este campo é ou não um campo obrigatório.
AcoesDisponiveis_ID	uniqueidentifier	Campo que relaciona esta tabela com a tabela AcaoDisponivel.
Campo_ID	uniqueidentifier	Campo que relaciona registos desta tabela com ela mesma.

Tabela 7 - Tabela Campo

A tabela "Template" tem o objetivo de armazenar os dados relativos aos *templates* criados pelos utilizadores. Para cada *template* é armazenado um XML, onde estão definidos os passos relativos ao *template* em questão.

Template		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
Nome	nvarchar(max)	Campo que armazena o nome dado pelo utilizador ao *template*.
TipoProcesso	int	Campo onde é armazena o tipo de processo em questão. Se é um processo civil ou processo penal.
XML	nvarchar(max)	Campo onde se vai guardando todos os dados relativos a um determinado *template* e aos seus respetivos passos.
DataCriacao	datetime	Data em que foi criado o template.
DataUltimaActualizacao	datetime	Data em que o *template* foi atualizado pela última vez.
Finalizado	Bit	Campo que indica se o *template* está pronto para ser utilizado.

Tabela 8 - Tabela Template

A tabela “Documento” serve para armazenar ficheiros relativos a um terminado *template*, de modo a ser possível descarrega-los quando a formação estiver a decorrer.

Documento		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
FileName	nvarchar(max)	Nome do documento guardado.
ContentType	nvarchar(max)	Campo que armazena o MIME *type* relativo ao ficheiro.
File	varbinary(max)	Campo que armazena o *array* de *bytes* relativo ao ficheiro.
ContentLength	int	Campo que armazena o tamanho do ficheiro em *bytes*.
Template_ID	uniqueidentifier	Campo que relaciona os documentos com um determinado *template*.

Tabela 9 - Tabela Documento

A tabela “ConfiguracaoPerfisTemplate” tem o objetivo de guardar as configurações acerca dos perfis que vão estar associados a um determinado *template* para uma formação. Nesta tabela é guardado um XML composto com os passos que constituem o *template*, mas agora cada um dos passos deve estar configurado.

ConfiguracaoPerfisTemplate		
Campo	**Tipo**	**Descrição**
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
Nome	nvarchar(max)	Campo que armazena o nome relativo ao nome dado pelo utilizador à configuração guardada.
XML	nvarchar(max)	Campo onde se guarda o XML relativo ao *template* mas agora com as novas configurações associadas a cada passo.
DataCriacaoOrUpdate	datetime	Data relativa à criação desta configuração.

Template_ID	uniqueidentifier	Campo que relaciona esta configuração com um determinado *template*.

Tabela 10 - Tabela ConfiguraçãoPerfisTemplate

A tabela "Instancia" armazena dados relativos a cada uma das formações. Com esta tabela é possível acompanhar em tempo real o passo em que se encontra a formação, bem como a data de início, a data de fim e a data do último passo realizado. Nesta tabela existe também um campo XML onde apenas é guardado um mapeamento entre o passo e o utilizador que deve realizar esse passo. Para terminar temos o campo ordem que indica qual a ordem do passo atual, e o local que é um campo em texto livre para mencionar o local onde decorre a formação.

Instancia

Campo	Tipo	Descrição
Id	uniqueidentifier	Campo que identifica cada registo.
Nome	nvarchar(max)	Nome dado pelo utilizador à formação arranca.
ProcessoId	uniqueidentifier	Identificador do processo no SIJ relativo à formação em questão.
PassoActual	uniqueidentifier	Identificador da tarefa que se encontra a ser realizada.
Ordem	int	Ordem no *template* relativamente à tarefa que se encontra a ser realizada.
XML	nvarchar(max)	Documento XML relativo ao mapeamento dos utilizadores com as tarefas que estes devem realizar.
DataInicio	datetime	Data em que a formação começou.
DataFim	datetime	Data em que a formação terminou.
UltimaActualizacao	datetime	Data da última tarefa realizada.

Local	nvarchar(max)	Local onde se realiza a formação.
ConfiguracaoPerfisTemplate_ID	uniqueidentifier	Campo que relaciona esta tabela com a tabela onde é feita a configuração dos perfis para as tarefas de um dado *template*.

Tabela 11 - Tabela Intancia

Para finalizar temos a tabela "EstadoPorPasso" que tem o objetivo de armazenar de uma forma mais detalhada a monitorização de cada passo. Aqui é armazenado o passo monitorizado, o utilizador que realizou o passo, bem como a data em que começou o passo e a data em que foi terminado. Temos ainda dois campos responsáveis por armazenar o resultado da execução do passo realizado pelo formando.

EstadoPorPasso

Campo	Tipo	Descrição
ID	uniqueidentifier	Campo que identifica unicamente cada registo.
UserId	uniqueidentifier	Identificador do utilizador que realizou este passo.
PassoId	uniqueidentifier	Identificador do passo a ser monitorizado.
DataInicio	datetime	Data em que o passo foi começado.
DataFim	datetime	Data em que o passo foi terminado.
Status	Bit	Booleano que indica se o passo foi ou não realizado com sucesso.
StatusInfo	nvarchar(max)	Mensagem para apresentar ao utilizador relativamente ao passo realizado.
Instancia_Id	uniqueidentifier	Campo que relaciona cada passo com uma formação.

Tabela 12 - Tabela EstadoPorPasso