



Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia Informática

Unidade Curricular de Laboratórios de Informática IV

Ano Lectivo de 2015/2016

Assistente para detetive

Alexandre Lopes Mandim da Silva (A73674)

Francisco José Torres Ribeiro (A70712)

Rafael Mota Oliveira (A72307)

Rui António Ramada Rua (A71418)

Grupo 3

Maio, 2016

L|4

Data de Recepção	
Responsável	
Avaliação	
Observações	

Assistente para detetive

Alexandre Lopes Mandim da Silva (A73674)

Francisco José Torres Ribeiro (A70712)

Rafael Mota Oliveira (A72307)

Rui António Ramada Rua (A71418)

Grupo 3

Maio, 2016

Resumo

O presente relatório descreve as três fases do projeto “Assistente para Detetive” que surgiu no âmbito de um contacto por parte da empresa “BragaDetetives” que procurava uma solução de software para agilizar o seu processo de trabalho.

A primeira fase consiste na fundamentação do projeto, sendo esta iniciada pela contextualização e apresentação do caso de estudo. Nestes dois tópicos, descrevemos genericamente o negócio do cliente e realizamos uma apresentação mais detalhada da solução informática.

Na motivação e objetivos são expostas as falhas nos métodos de trabalho da agência e os objetivos a alcançar no término da construção desta solução.

Seguidamente, procedeu-se à elaboração de uma maquete que representa a arquitetura do sistema idealizado, evidenciando assim, de forma simplista, os constituintes do sistema e a sua interação.

Foi também feito um planeamento das tarefas que constituem as três fases do projeto com base no tempo que prevemos ser necessário para as realizar, tendo este processo sido auxiliado pela construção de um Diagrama de *Gantt*.

Terminada a fase de fundamentação, é apresentada a fase de especificação em que são especificados e analisados os requisitos do sistema em questão. Nesta mesma fase, além do levantamento e da análise de requisitos, foram também criados os modelos de sistema (em UML). Estes modelos consistem no diagrama de *Use Cases*, diagramas de Sequência, diagramas de Atividades, modelo de Domínio e diagrama de Classes.

Com os requisitos levantados e analisados e os modelos construídos procedeu-se à idealização e elaboração da base de dados a partir da construção dos respetivos modelos conceptual e lógico, baseados nos requisitos.

Para concluir a segunda fase, são apresentados os *mockups* que constituem o planeamento da *interface* com o utilizador.

Na fase de implementação, que constitui a fase final do projeto, são apresentadas as funcionalidades que foram implementadas no sistema, podendo também ser vistas algumas imagens que ilustram o seu funcionamento.

Área de Aplicação: Desenvolvimento de um sistema informático para uma agência de detetives.

Palavras-Chave: Desenvolvimento de Software, Engenharia de Software, Aplicação, Análise de Requisitos, Metodologias de Software, Modelo em Cascata, Bases de Dados Relacionais,

Assistente de Campo, *Back Office*, *Front Office*, Investigação, Detetives, Gestão de Projeto,
Maquete, Diagrama de *Gantt*, Microsoft Project.

Índice

Resumo	i
Índice	iii
Índice de Figuras	vi
Índice de Tabelas	ix
1. Introdução	1
1.1. Contextualização	1
1.2. Apresentação do Caso de Estudo	2
1.3. Motivação	3
1.4. Objetivos	4
1.5. Maquete	6
1.6. Planeamento e gestão do projeto	7
1.7. Estrutura do relatório	8
2. Requisitos	10
2.1. Requisitos de Utilizador	10
2.2. Requisitos de Sistema	12
2.2.1. Funcionais	12
2.2.2. Não Funcionais	16
3. Modelos de Sistema	17
3.1. Diagramas UML	17
3.1.1. Diagramas de Use Case	17
3.1.1.1. Diagrama de Use Case - Back-Office	17
3.1.1.1.1. Exemplo de especificação de Use Case - Criação de uma tarefa	18
3.1.1.2. Diagrama de Use Case - Front-Office	20
3.1.2. Diagramas de sequência	24
3.1.2.1. Exemplo de Diagrama de Sequência- Criação de uma tarefa	24
3.1.2.2 Exemplo de Diagrama de Sequência- Exportar para Front-Office	25
3.1.2.3. Exemplo de Diagrama de Sequência- Realizar tarefa	26
3.1.3 Diagramas de Atividade	29
3.1.3.1. Exemplo de diagrama de atividade – Criação de uma tarefa	29
3.1.3.2. Exemplo de diagrama de atividade – Exportar para Front-Office	32
3.1.3.3. Exemplo de diagrama de atividade – Realizar tarefa	33
3.1.4. Modelo de Domínio	37
3.1.5. Diagrama de Classes	39
4. Base de Dados	40
4.1. Modelo Conceitual	40
4.1.1. Identificar os tipos de entidades	40

4.1.2. Identificar tipos de relacionamento	40
4.1.3. Identificar e associar atributos com os tipos de entidades e relacionamentos	
42	
4.1.4. Determinar domínios dos atributos	44
4.1.5. Determinar chaves primárias, candidatas e alternativas	45
4.1.6. Desenho do diagrama ER	46
4.1.7. Revisão do modelo de dados com o utilizador	46
4.2. Modelo Lógico	46
4.2.1. Passagem do modelo conceptual para o modelo lógico	47
4.2.2. Validação do modelo lógico através de normalização	49
4.2.3. Elaboração e validação do esquema lógico da base de dados	49
4.2.4. Definição do tamanho inicial da base de dados e análise do seu crescimento futuro	50
4.2.5. Revisão do modelo lógico final com os futuros utilizadores	50
4.3. Modelo físico	50
4.3.1. Acesso à base de dados	50
4.3.2. <i>Stored Procedures</i>	52
4.3.3. <i>Views</i>	55
5. Mockups	57
6. Implementação	63
6.1. <i>BackOffice</i>	63
6.1.1. Namespaces	63
6.1.2. Conexão à base de dados (<i>EntityFramework</i>)	63
6.1.3. Interface	65
6.1.4. Funcionalidades	69
6.2. <i>FrontOffice</i>	75
6.2.1. Ferramentas utilizadas	75
6.2.2. Menu Principal	76
6.2.3. Persistência de dados	77
6.2.4. Ver tarefas	79
6.2.5. Importação de tarefas	79
6.2.6. Exportação de tarefas	80
6.2.7. Tarefa	81
6.2.8. Indicar caminho	83
6.2.9. Tirar Fotografia	85
6.2.10. Registar Nota de Voz	86
6.2.11. Registar Nota Escrita	88
6.2.12. Obter informações	89
6.2.13. Concluir Tarefa	89
6.2.14. Suspender Tarefa	90

7. Balanço do planeamento e gestão do projeto	92
8. Conclusões e trabalho futuro	93
Referências	95
Lista de Siglas e Acrónimos	96
Anexos	97
I. Anexo 1 – Exemplos de especificações de Use Case	98
II. Anexo 2 – Exemplos de diagramas de sequência	100

de índice.

Anexos

I. Anexo 1	98
------------	----

Índice de Figuras

Figura 1 Maquete do sistema "Assistente de Detetive"	6
Figura 2 - Diagrama de <i>Gantt</i>	7
Figura 3 Diagrama de Use Cases - BackOffice	17
Figura 4 Especificação do Use Case "Criação de uma tarefa"	19
Figura 5 Especificação do Use Case "Exportar para o frontOffice"	20
Figura 6 Diagrama de Use Case - FrontOffice	21
Figura 7 Especificação do Use Case "Realizar tarefa"	23
Figura 8 Diagrama de Sequência "Criação de uma tarefa"	25
Figura 9 Diagrama de Sequência "Exportar para o frontOffice"	26
Figura 10 Diagrama de Sequência "Visualizar plano de atividades"	27
Figura 11 Diagrama de Sequência "Realizar tarefa"	28
Figura 12 Diagrama de Atividade "Criação de uma tarefa"	30
Figura 13 Descrição da ação "Pede informação caraterizadora"	31
Figura 14 Descrição da ação "Valida informação"	31
Figura 15 Diagrama de Atividade "Exportar para frontOffice"	32
Figura 16 Descrição da ação "Verifica quais as tarefas a ser enviadas"	33
Figura 17 Diagrama de Atividade "Realizar tarefa"	35
Figura 18 Descrição da ação "Pesquisa caminho local investigação"	36
Figura 19 Descrição da ação "Escolher o que pretende pesquisar"	36
Figura 20 Descrição da ação "Pesquisa localização"	36
Figura 21 Modelo de Domínio	37
Figura 22 Diagrama de Classes	39
Figura 23 Modelo Conceptual	46
Figura 24 Surgimento de novas tabelas	48
Figura 25 Modelo Lógico	49
Figura 26 - Criação de <i>logins</i>	51
Figura 27 - Criação de <i>users</i>	51
Figura 28 - Criação do Stored Procedure AgentesEnvolvidos	53
Figura 29 - Criação do Stored Procedure CasosAtivos	53
Figura 30 - Criação do Stored Procedure dadosRelatorioAgentes	54
Figura 31 – Criação do Stored Procedure tarefasDosAgentesDeUmIC	54
Figura 32 - Criação do Stored Procedure TarefasIC	54
Figura 33 - Criação do Stored Procedure TarefasNaoSincronizadas	55
Figura 34 - Criação da view viewEquipas	56
Figura 35 - Criação da view viewInspectoresChefe	56
Figura 36 Mockup Criar Tarefa	57
Figura 37 Mockup Ver Casos	58

Figura 38 Mockup Visualizar Equipa	59
Figura 39 Mockup Menu frontOffice	59
Figura 40 Mockup Ver Tarefas	60
Figura 41 Mockup Realizar Tarefa	61
Figura 42 Mockup Obter informações complementares	62
Figura 43 - Utilização de um Entity Data Model	64
Figura 44 - Código <i>login</i>	64
Figura 45 - Janela <i>login</i>	65
Figura 46 - Menu Diretor	66
Figura 47 - Menu IC	66
Figura 48 - Janela referente a criar uma tarefa	67
Figura 49 - Janela Menu e janela visualizar equipas	68
Figura 50 - Janela Menu e janela verificar tarefas	68
Figura 51 - Utilização de <i>RadioButton</i>	68
Figura 52 - Exemplo <i>DataGrid</i>	69
Figura 53 - Tratamento <i>ComboBox</i> (<i>Criar tarefa</i>)	70
Figura 54 - Código referente à criação de uma tarefa	71
Figura 55 - Classe TEmail	72
Figura 56 - Definição das palavras-chave	73
Figura 57 - Reconhecimento das palavras do ficheiro audio	73
Figura 58 - Implementação da criação de documento <i>XML</i>	74
Figura 59 - Exemplo XML	75
Figura 60 Menu principal <i>FrontOffice</i>	77
Figura 61 Exemplo de um ficheiro XML que representa o estado do FrontOffice	78
Figura 62 Ver tarefas	79
Figura 63 Excerto de código que permite estabelecer conexão ao BackOffice	80
Figura 64 Excerto de código que permite estabelecer ligação para exportar para o BackOffice	81
Figura 65 Ecrã principal de uma tarefa	82
Figura 66 Exemplo de preenchimento das informações caracterizadoras de uma tarefa	82
Figura 67 Obtenção da localização atual	84
Figura 68 Indicar localizações	85
Figura 69 Funcionalidade de Tirar Fotografia	86
Figura 70 Implementação do registo de uma nota de voz	86
Figura 71 Processo de paragem da gravação de voz	87
Figura 72 Registo de nota de voz	87
Figura 73 Registo de uma nota escrita	88
Figura 74 Funcionalidade Obter informações	89
Figura 75 Tarefa a aguardar conclusão	90

Figura 76 Caixa de diálogo - Conclusão de tarefa	90
Figura 77 Opções para suspensão de uma tarefa	91

Índice de Tabelas

Tabela 1 Tabela de entidades	40
Tabela 2 Tabela de relacionamentos	41
Tabela 3 Tabela de atributos de entidades e relacionamentos	43

1. Introdução

No primeiro ponto deste capítulo procedemos à apresentação do contexto do projeto, onde é descrito o negócio do cliente de uma forma generalizada e em que se baseiam as atividades realizadas pelos agentes. De seguida, na secção “Apresentação do Caso de Estudo” é feita uma caracterização mais detalhada da solução informática que se pretende realizar de forma a satisfazer as necessidades do cliente, necessidades estas que foram mencionadas na secção anterior “Contextualização”.

As duas secções seguintes são a “Motivação” e “Objetivos”. Na primeira são enumeradas e narradas ao detalhe as falhas presentes nos métodos utilizados pela empresa. Finalmente, na secção “Objetivos” são descritas as metas que este grupo se propõe a atingir de modo a que a conclusão deste projeto se traduza numa boa peça de software capaz de responder a todos os requisitos do cliente.

Neste capítulo introdutório pode também ser vista uma maquete que ajuda a ilustrar de uma forma simplificada o sistema, bem como uma descrição do processo de planeamento do projeto e justificações para as decisões tomadas na gestão do mesmo.

1.1. Contextualização

A realização deste projeto surge da necessidade de uma agência de detetives em melhorar o acompanhamento no terreno dos seus agentes.

Esta empresa procura um sistema informático que dê um auxílio constante no terreno, aos seus agentes, aquando da execução das tarefas a eles predestinadas.

As tarefas dos detetives envolvem, numa grande maioria das vezes:

- Ações de vigia;
- Investigação de eventos (não necessariamente crimes);
- Obtenção de algum tipo de objeto (provas, por exemplo);
- Espionagem empresarial;
- Investigação de assassinatos;
- Investigação de fraudes em seguradoras;
- Problemas familiares (uso de drogas, adultério, etc.);
- Segurança.

Estas tarefas são, quase na totalidade, baseadas em deslocações programadas a determinados locais. Como tal, os detetives necessitam de se fazer acompanhar de um dispositivo de fácil portabilidade que os possa auxiliar na recolha de provas e no relato dos acontecimentos que ocorram durante as suas tarefas de investigação.

O sucesso destas atividades passa por uma disciplinada organização da sequência das tarefas a executar, registo dos detalhes de acontecimentos relacionados e correta catalogação (armazenamento) de toda a informação recolhida no decorrer das missões.

Para além do mencionado, no término das missões, é necessária a criação de um relatório final que documente a investigação realizada. A elaboração desse relatório passa por uma análise completa de toda a informação recolhida pelo detetive. Assim sendo, a correta organização dessa informação auxilia no processo de escrita do documento final.

Como tal, o sistema informático a ser desenvolvido deve satisfazer as necessidades mencionadas.

1.2. Apresentação do Caso de Estudo

Como mencionado no ponto anterior, é necessário criar uma solução informática que responda às necessidades transmitidas pela agência de detetives ‘BragaDetetives’.

Esta solução, resumidamente, deve permitir que cada agente siga um plano de atividades previamente definido, recolha e armazene dados referentes a cada uma das atividades e, por fim, centralizar toda a informação obtida pelos diversos agentes num *back office*.

A solução referida divide-se em duas aplicações:

- Assistente de campo (*Front Office*; instalado no dispositivo móvel);
- *Back office* (ponto de sincronização da informação recolhida pelos agentes).

Genericamente, um assistente de campo deve auxiliar o utilizador nas suas tarefas laborais. Como tal, esta peça deve permitir ao utilizador não só consultar o plano de atividades que deve executar, como também guardar informação sobre os eventos ocorridos. Para além disso, outra forma de auxílio fornecida pelo assistente consiste também em orientar geograficamente o utilizador no seu percurso.

Posta esta definição, o assistente de campo idealizado para a agência de detetives em questão deve:

- Permitir configurar o modo de atuação;
- Permitir consultar o plano de atividades a executar (previamente agendadas);
- Registar informações complementares sobre uma determinada atividade. Este registo pode passar por fotografias, gravações de voz e/ou coordenadas geográficas.
- Guiar geograficamente o agente durante o seu percurso no terreno;
- Fornecer informação complementar sobre um determinado tópico relacionado com o trabalho a ser realizado;
- Sincronizar/exportar a informação para o *back office*.

O *back office* é um programa que se encontra instalado nos escritórios da empresa e deve ser capaz de:

- Definir os planos de atividades das missões dos agentes. Mais concretamente, explicitar as tarefas que devem ser executadas em cada investigação e os locais a que o detetive se deve deslocar;
- Permitir que os Assistentes de Campo importem os planos de atividade;
- Transformar as gravações de voz guardadas pelo agente em documentos escritos e anotados;
- Manter numa base de dados relacional todos os registos efetuados pelos agentes durante o seu trabalho de campo;
- Visualizar os relatórios produzidos e organizados por detetive e investigação;
- Enviar os relatórios por *email* a outros agentes.

Assim sendo, é esperado que cada uma destas aplicações implemente as respetivas funcionalidades e que juntas sejam capazes de se complementar, constituindo assim um sistema global que solucione o problema proposto pela empresa em questão.

Foi efetuada também uma pesquisa com o objetivo de encontrar soluções informáticas que resolvessem o problema que nos foi proposto ou até que possuíssem funcionalidades similares que pudessem substituir, de alguma forma, a nossa solução. Após essa pesquisa, concluímos que não existe nenhuma aplicação específica para utilizar na área do nosso projeto que forneça todas as funcionalidades que a nossa solução oferecerá. Contudo, existem várias aplicações que oferecem algumas funcionalidades previstas no nosso sistema e que podem ser utilizadas para solucionar problemas que a nossa solução procura resolver, mas cuja área de aplicação não é concretamente a mesma do nosso projeto.

Uma das aplicações que encontrámos foi a aplicação *Zillow*, uma aplicação com foco no mercado imobiliário (e não no negócio de investigação de detetives) e que tem uma vasta informação sobre os espaços que estão anunciados no site, tal como número de divisões, plantas e valores do edifício, entre outras. De acordo com o site, cuja referência se encontra disponibilizada na parte do relatório destinada para o efeito, trata-se de uma ótima aplicação para os detetives usarem de maneira a efetuar pesquisas sobre os espaços que queiram investigar.

1.3. Motivação

O motivo que levou à realização deste projeto consistiu no facto de existirem problemas, por parte da agência “BragaDetetives”, não só nos métodos de recolha da informação como também na organização desta.

O processo de recolha de dados durante as investigações revelava falhas, uma vez que eram utilizados métodos convencionais que se revelavam pouco eficazes. Um exemplo destes métodos é o suporte escrito a que os agentes recorriam quando queriam registar notas rápidas. O registo por gravação de voz destas notas é um processo que se revela mais eficaz, menos

complexo e mais cómodo (o detetive não necessita de transportar material de escrita, por exemplo).

Outra lacuna detetada nos processos de investigação dos detetives consiste na extensa descrição escrita que os agentes tinham de efetuar de eventos que presenciavam, o que se revelava uma tarefa bastante fustigante e demorada, podendo comprometer a correta descrição do evento à medida que ele ocorre. Para além disso, nem todos os acontecimentos ou objetos são suscetíveis de ter uma explicação escrita que faça justiça à sua verdadeira natureza. Deste modo, a melhor forma de registar este tipo de ocorrências seria algum tipo de registo visual, como por exemplo uma fotografia. Para além de ser um modo de registo de acontecimentos muito mais rápido, as fotografias fornecem a possibilidade de visualizar novamente, e com muito mais detalhe, as ocorrências documentadas. Posto isto, com a solução idealizada será permitido aos detetives tirar fotografias de assuntos que se revelem dignos de tal registo.

As missões dos detetives são baseadas em deslocações. Por vezes, o que acontece é que um agente pode não saber como alcançar um determinado local que se encontra especificado no seu plano de atividades. Face a este problema, existe a necessidade de acompanhamento do percurso dos agentes através de algum tipo de orientação geográfica que a aplicação possa fornecer (exemplo: indicação do trajeto a percorrer para chegar a um destino).

A informação recolhida durante a missão está muitas vezes associada a um determinado local. A necessidade de interligar estes registos com a respetiva localização geográfica é outro dos motivos que levou à realização deste trabalho.

Quando um agente realiza uma dada atividade pode vir a ser necessário obter rapidamente informações complementares sobre um dado assunto. Por exemplo, foi relatado pelos agentes que, muitas vezes, enquanto observavam suspeitos, estes entravam em estabelecimentos cujo foco de atividade lhes era desconhecido. Face a esta situação os agentes encontravam-se constantemente inseguros e desinformados o que poderia comprometer o modo de atuação. Como tal, os detetives expressaram a vontade de possuir algum tipo de ferramenta que lhes possibilitasse a obtenção imediata de informação e consequentemente prosseguir a missão com maior segurança.

Do ponto de vista do trabalho nos escritórios da agência, a informação relativa às investigações encontrava-se armazenada de forma dispersa em vários portfólios localizados em diversos compartimentos. Este método revela-se desorganizado, ineficiente e pouco escalável, o que cria dificuldades em encontrar e cruzar dados à medida que a quantidade de informação aumenta. Este constituiu outro dos motivos que levou à idealização de uma plataforma capaz de armazenar e gerir de forma centralizada e mais segura todos estes dados.

1.4. Objetivos

Globalmente, a solução informática apresentada visa facilitar o trabalho de campo dos detetives e a organização da informação na agência, tornando estes processos mais cómodos, rápidos e eficazes.

Com a possibilidade de captura de fotografias e de gravação de voz, pretendemos aumentar a rapidez do processo de relato dos acontecimentos. Estes relatos eram realizados através de escrita em papel, pelo que um dos objetivos a alcançar com esta nova funcionalidade é a substituição deste suporte, que era o responsável por alguma falta de eficiência. Através da adoção deste método procuramos que os detetives sejam mais eficazes, produtivos e ágeis na realização do trabalho de campo diário.

A orientação geográfica disponibilizada pelo assistente de campo tem o objetivo de possibilitar ao detetive programar o seu percurso e até mesmo evitar um desvio. Como tal, esta funcionalidade permite não só fornecer ao detetive o conhecimento do mapa geral, mas também conduzi-lo pelo caminho mais adequado. O tempo é um recurso precioso no trabalho de um agente e, desta forma, evita-se a perda de tempo na escolha de um trajeto, uma vez que este é calculado pela aplicação. Além disso, outro objetivo é fornecer o tempo médio que o agente demora a percorrer um dado percurso, uma vez que pode ser fundamental para a tomada de decisões.

A implementação de um sistema GPS na aplicação não tem só propósitos de orientação. Outra meta bastante importante é o armazenamento das informações relativas às deslocações do agente à medida que este se desloca no terreno.

Com a criação de uma ferramenta que possibilite a obtenção imediata de informação complementar sobre um assunto especificado pelo detetive pretende-se que este possa tomar decisões mais sustentadas, aumentando, desta forma, a possibilidade de ter êxito nas suas tarefas.

Possibilitar uma atribuição do plano de tarefas de cada agente, de forma automatizada e livre de falhas, é outro dos aspetos a alcançar por parte do *back office*. Esta implementação procura a redução do trabalho de distribuição das tarefas, anteriormente feito por um trabalhador, aumentando a produtividade e reduzindo custos no escritório. Para além disso, este método garante a segurança e privacidade da informação das investigações que é apenas destinada aos “olhos” do respetivo detetive.

Por fim, a construção de uma base de dados relacional capaz de suportar toda a informação sobre os agentes e as suas atividades de forma centralizada é outro dos objetivos desta solução. Desta forma, este método visa aumentar a rapidez, eficiência e segurança no acesso e cruzamento da informação, para além de evitar a dispersão dos documentos que contenham os dados.

1.5. Maquete

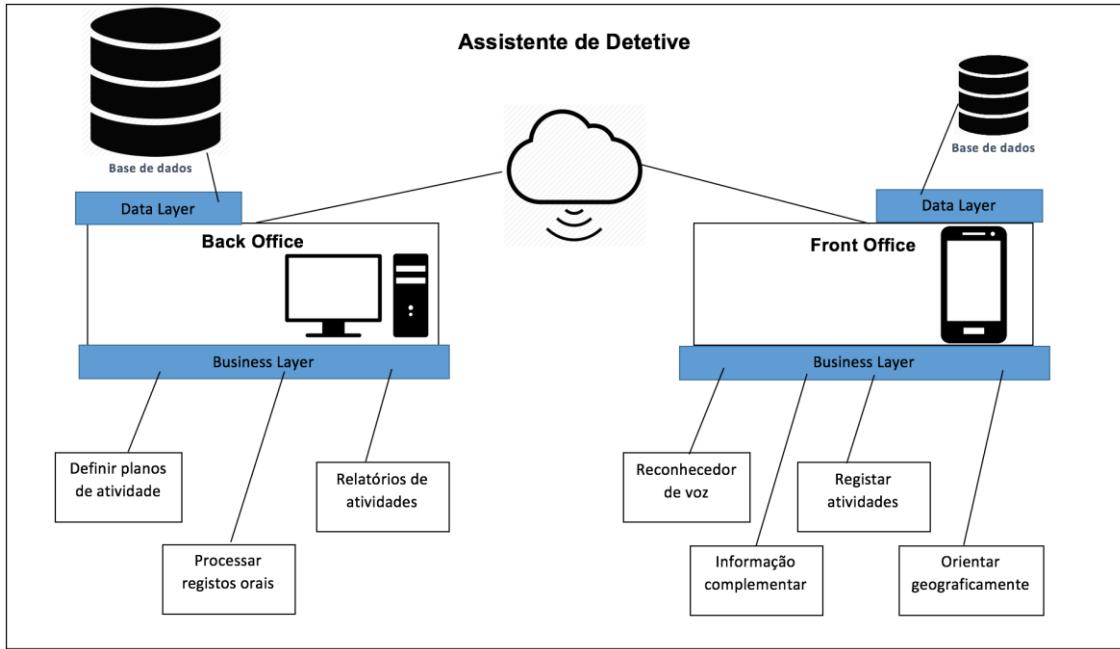


Figura 1 Maquete do sistema "Assistente de Detetive"

O assistente de detetive é composto por dois sistemas: *back office* e *front office*.

O *back office* consiste numa aplicação, em permanente atividade, situada num dos computadores da empresa. Esta aplicação, tal como mostrado na figura 1, é responsável pela definição dos planos de atividade dos agentes, criação de relatórios relativos às atividades efetuadas e processamento dos registos orais captados pelos agentes, transformando-os em documentos escritos anotados (*XML*). O *back office* é o responsável pelo armazenamento e coerência de toda a informação, mantendo-a numa base de dados relacional.

O *front office* (Assistente de Campo) é a aplicação instalada em cada *smartphone* pertencente aos agentes. Esta aplicação fornece a cada agente os seus planos de atividade, permite acrescentar informação sobre uma dada tarefa (notas de voz, fotografias e localizações geográficas), orienta o agente geograficamente numa dada investigação e permite pesquisar informações acerca de qualquer assunto que o agente pretenda. Os dados referentes às informações de uma dada atividade e às pesquisas efetuadas são guardados numa "pequena" base de dados relacional no *smartphone*.

Quando o agente se conecta ao *back office* (sede da empresa) é feita a sincronização entre estes dois sistemas. O *front office* envia as informações recolhidas sobre as atividades realizadas e recebe novos planos de atividade, caso existam. O *back office* processa toda a informação recebida, guardando-a na base de dados.

1.6. Planeamento e gestão do projeto

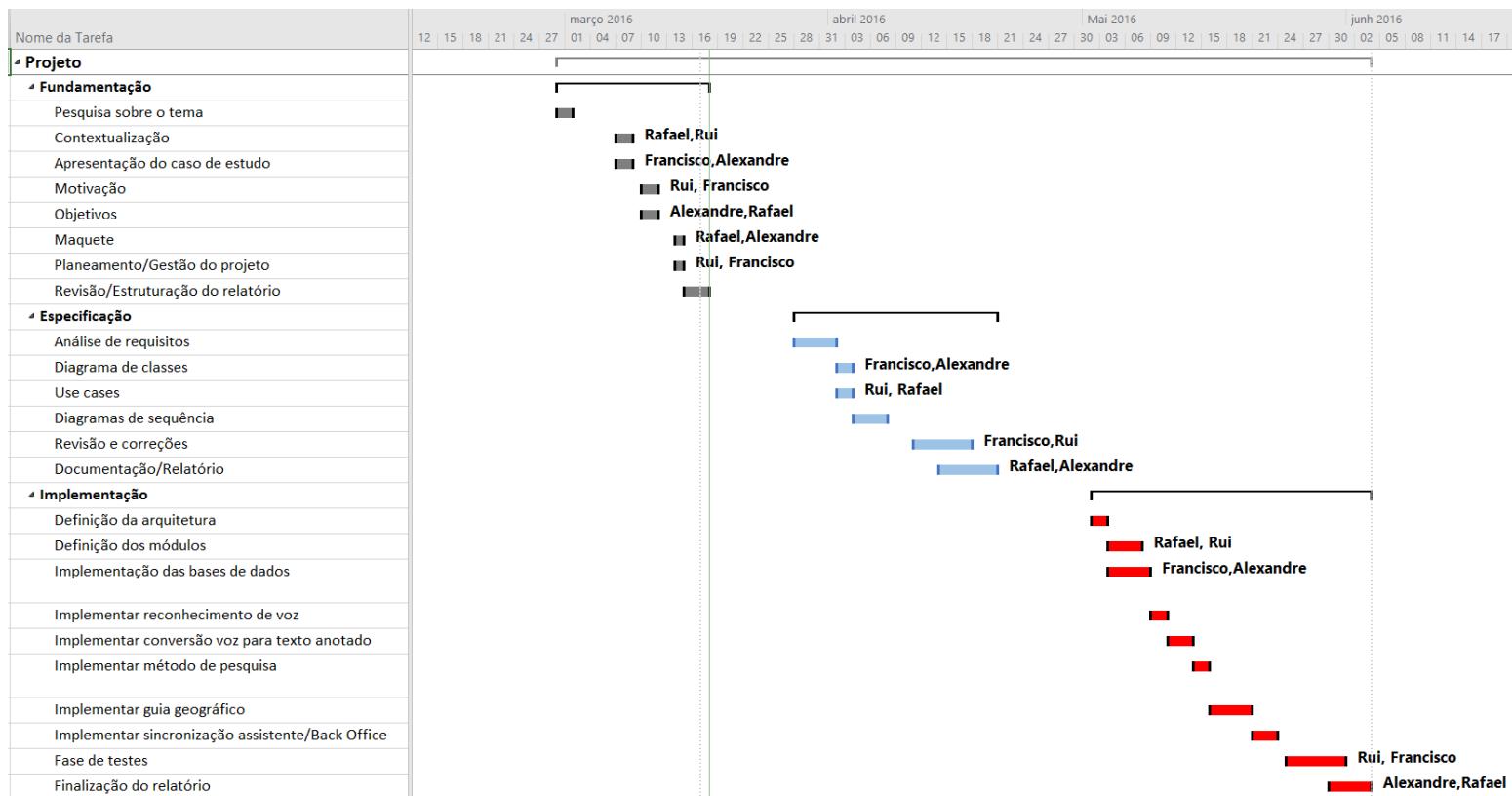


Figura 2 - Diagrama de Gantt

A primeira fase do projeto passou pela fundamentação do tema. Primeiramente, foi efetuada uma intensiva pesquisa acerca deste mesmo assunto com o objetivo de recolher informação que auxiliasse o grupo a inserir-se no contexto do que é o trabalho de um detetive. Esta pesquisa permitiu ter uma percepção dos aspetos inerentes às atividades realizadas pelos agentes, o que possibilitou sustentar a fundamentação do projeto. Para além disso, este estudo também nos permitiu percecionar em que medida é que o projeto iria facilitar a tarefa dos detetives.

Partindo das informações recolhidas, concebemos grande parte do relatório (contextualização, apresentação do caso de estudo, motivação e objetivos). A maquete (que consiste na idealização da arquitetura da aplicação) e a presente secção, foram realizadas de forma paralela, o que implicou a divisão do grupo em duas sub-equipas. Após a realização das tarefas correspondentes a esta fase do projeto, foram planeadas e delineadas as restantes fases que irão compor a realização do projeto, nomeadamente a modelação e a implementação.

Relativamente a cada tarefa, foi alocado para cada uma um período de tempo que achamos adequado para a realização da mesma, não sendo possível prever com exatidão o período de desenvolvimento de cada uma delas. Imaginámos que de forma a cumprir os prazos de entrega seja necessário realizar algumas das tarefas previstas simultaneamente. São

previstos também períodos de tempo em que não se realiza qualquer atividade relacionada com o projeto. Estas paragens relacionam-se com a necessidade de reservar algum espaço no calendário para que os elementos do grupo possam desenvolver outros projetos.

Na fase de modelação, o grupo prevê que seja necessário um trabalho simultâneo dos quatro elementos constituintes para a análise de requisitos, sendo que ficaram previstos, aproximadamente, 5 dias para a realização dessa tarefa. De seguida, passar-se-á à conceção dos diagramas UML necessários à idealização do sistema, finalizando-se esta fase com um conjunto de revisões e correções. Julgamos que esta última tarefa será outra das partes em que teremos de investir mais tempo, de forma a garantir a coerência entre os diagramas construídos.

Na última fase do projeto, correspondente à fase de implementação do sistema, foi onde tivemos maior dificuldade em prever o tempo que viria a ser despendido em cada tarefa. Contudo, julgámos que a implementação das bases de dados será uma das tarefas que levará mais tempo a ser completada. Relativamente às outras tarefas, o principal obstáculo que se opõe à capacidade de previsão do tempo é a falta de experiência por parte do grupo no uso das ferramentas e mecanismos que irão ser utilizados para a sua realização. Tal como na fase anterior do projeto, para a tarefa de testes e correção de erros foi alocado um período considerável de forma a garantir o correto funcionamento das funcionalidades implementadas. É previsto que a finalização do relatório se inicie (através de uma divisão em sub-grupos) antes da parte de testes terminar, de maneira a garantir que não só todo o processo de descrição de desenvolvimento do projeto esteja revisto e em conformidade com o que irá ser apresentado, mas também que o relatório seja acompanhado por documentação suficientemente aprofundada, clara e organizada.

Por último, é de realçar que a divisão das tarefas por elementos do grupo foi feita através de uma discussão em equipa em que foram tidos em conta os pontos fortes de cada pessoa. Contudo, esta atribuição inicial de responsabilidades é suscetível de ser alterada com base em aspetos futuros que o justifiquem, como por exemplo, dificuldades de execução de determinada tarefa que suscite necessidade de cooperação entre os vários membros do grupo.

Um atraso no desenvolvimento do projeto poderá ser outro fator que leve a uma redistribuição dos trabalhos e, para além disso, a uma alteração no tempo destinado a cada tarefa. Além disso, uma coordenação entre os elementos da equipa poderá ser necessária como forma de garantir a coesão e integração de todo o trabalho até então realizado.

1.7. Estrutura do relatório

O presente relatório está estruturado em vários capítulos de maneira a agrupar vários assuntos de interesse.

O segundo capítulo consiste nos requisitos do sistema e é onde são apresentados e analisados os requisitos de utilizador, bem como os requisitos de sistema que consistem numa apresentação mais detalhada e técnica dos requisitos de utilizador. Os requisitos de sistema encontram-se também divididos nos requisitos funcionais e nos não funcionais.

No terceiro capítulo são apresentados e explicados os modelos do sistema idealizado. Neste capítulo podem ser vistos alguns diagramas *UML* e várias especificações de *Use Cases*.

No quarto capítulo procede-se à apresentação da base de dados. É neste capítulo que são apresentados os modelos conceputal, lógico e físico que permitem ter uma compreensão completa da base de dados construída, bem como dos passos que levaram à sua idealização e criação.

No quinto capítulo podem ser visualizados os *mockups* que constituem a idealização da *interface* que o sistema terá para com o utilizador.

No sexto capítulo procede-se à apresentação da parte de implementação do projeto. Nesta parte elaboram-se, de forma separada, as implementações do subsistema *BackOffice* e do subsistema de *FrontOffice*.

No sétimo capítulo faz-se um balanço sobre o planeamento e gestão do projeto efetuado na fase inicial do trabalho, comentando-se o cumprimento e o não cumprimento dos vários aspectos que foram planeados inicialmente.

No final, pode ser lida a conclusão do trabalho com algumas considerações sobre o projeto que foi realizado, bem como alguns comentários sobre o trabalho futuro.

2. Requisitos

O levantamento e análise de requisitos é uma das fases mais importantes na especificação de um sistema de software. A fim de realizarmos um bom levantamento de requisitos foram realizadas entrevistas a vários trabalhadores da empresa com diferentes cargos. Além disso foi realizado um trabalho de observação destes trabalhadores, ao longo de um dia laboral, com o intuito de recolher aspetos do problema que não sejam facilmente transmitidos através de comunicação direta, como as entrevistas.

Nos seções seguintes vamos apresentar os requisitos de utilizador e os de sistema, sendo que nestes últimos fazemos a distinção entre os funcionais e os não funcionais.

2.1. Requisitos de Utilizador

Na empresa *BragaDetetives* existem três tipos de entidades com responsabilidades distintas: o **Diretor**, **Inspetor-Chefe** e o **Agente**. Como tal, vão ter necessidades diferentes de interação com o sistema.

Para dar início à enumeração e explicação dos requisitos específicos das diferentes entidades, é importante perceber alguns conceitos globais do caso de estudo em questão:

1. Na empresa referida, as três entidades mencionadas formam uma hierarquia em que a entidade máxima é o Diretor, seguida do Inspetor-Chefe e finalmente pelo Agente;
2. Um caso corresponde a um pedido de investigação feito por um cliente à empresa. Face a este pedido são estipuladas tarefas que levem à resolução/concretização do caso. Além disso, um caso contém um nome, uma descrição, objetivos globais a alcançar e um espaço para considerações finais (relatório);
3. Uma tarefa consiste num objetivo a alcançar por parte do Agente. Uma tarefa é composta por um local de investigação, objetivo, descrição, título e o caso a que é referente;
4. Um plano de atividade é um conjunto de tarefas que o Agente deve executar. Este conjunto pode ser formado por tarefas de diversos casos;

Apresentados estes conceitos, seguem-se os vários requisitos referentes ao Diretor:

5. O Diretor é responsável por receber e avaliar os casos propostos pelos clientes da empresa e por entregá-los a um Inspetor-Chefe;

6. O Diretor é quem nomeia e supervisiona os Inspetores-Chefe;
7. Quando da contratação de um Agente, é o Diretor que cria o seu perfil de acesso ao sistema e o associa a uma equipa de agentes liderada por um Inspetor-Chefe;

Especificados os requisitos do Diretor procedem-se os requisitos associados a um Inspetor-Chefe:

8. Um Inspetor-Chefe é responsável por uma equipa de Agentes;
9. Cada Inspetor-Chefe pode visualizar apenas os casos que lhe estão associados. Com base nestes casos, o Inspetor-Chefe define planos de atividade (requisito 4) para a equipa de Agentes que coordena;
10. O Inspetor-Chefe é o responsável por dar como concluída a investigação/resolução de um dado caso a ele atribuído;
11. O Inspetor-Chefe pode atuar como um Agente, ou seja, pode definir planos de atividade para ele próprio;
12. Um Inspetor-Chefe pode enviar relatórios por email sobre casos de que é responsável;

Em baixo, são apresentados os requisitos que dizem respeito aos Agentes:

13. Cada Agente, caracterizado pelo seu código de identificação, nome próprio e apelido, apenas responde ao Inspetor-Chefe que o coordena;
14. Um Agente apenas pode visualizar informações sobre os casos que pertençam à equipa em que está inserido. Este também deve conseguir obter o plano de atividade, para si definido, pelo seu Inspetor-Chefe;
15. O dia-a-dia de um Agente é consultar o seu plano de atividades e cumpri-lo;
16. Com base numa tarefa, e caso o Agente pretenda, deve ser fornecido o percurso para chegar ao local pretendido;
17. Deve ser permitido aos Agentes tirar fotografias, notas escritas e notas em voz e anexar toda a informação a um local geográfico (coordenadas GPS). Toda esta informação está associada à respetiva tarefa;
18. Os Agentes devem ter a possibilidade de obter informações complementares de forma rápida sobre um determinado assunto.

Para finalizar, existem outros requisitos que correspondem às necessidades do sistema central:

19. Todas as informações recolhidas pelos Agentes devem ser exportadas para o sistema central. Este sistema corresponde ao *BackOffice*, o *frontOffice* corresponde ao sistema transportado pelos Agentes durante as tarefas de campo;

20. O sistema central deve traduzir cada registo de voz recolhido por um Agente no terreno para um documento escrito;
21. A informação de todos os casos deve ser guardada no sistema instalado na empresa.

2.2. Requisitos de Sistema

2.2.1. Funcionais

- 1.1. Como mencionado no requisito 1 existe uma hierarquia de entidades a ser respeitada. Isto traduzir-se-á em diferentes perfis de acesso que vão separar as funcionalidades e privilégios de cada utilizador do sistema;
 - 1.2. Para um utilizador ter acesso ao seu perfil e às funcionalidades do sistema, este tem que inserir as suas credenciais. Note-se que há uma exceção para o Diretor visto este não ter nenhum *login* previsto para o *frontOffice* uma vez que esta entidade não realiza trabalho de campo;
 - 1.3. Com base nos pontos 1.1 e 1.2 podemos concluir que é fundamental a implementação de uma autenticação no sistema por parte de todas as entidades. Esta autenticação seguir-se-á pela apresentação da interface com que o utilizador em causa irá interagir. As permissões de todas as funcionalidades são geridas pelo motor de base de dados, sendo este o responsável por decidir se o utilizador pode usufruir delas ou não.
-
- 2.1. O requisito 2 aponta para a existência de casos no *backOffice* que consistem numa correspondência com um pedido de investigação por parte de um dado cliente;
 - 2.2. O *backOffice* deve permitir, aquando da criação de um caso, atribuir-lhe um nome, uma descrição textual, objetivos/metas que devem ser atingidos durante a investigação em causa;
 - 2.3. O *backOffice* deve possibilitar a criação de tarefas (mencionadas no requisito 3) que se inserem no contexto de um caso, isto é, correspondem a ações/objetivos que devem ser cumpridos no âmbito de uma investigação de maneira a estabelecer uma ligação entre os casos e as tarefas que o constituem.
-
- 3.1. Para a criação de uma tarefa é fundamental que exista um conjunto de informações que a caracterize corretamente:
 - O **título** indica o assunto a que esta diz respeito;
 - A **descrição** indica, de forma clara, em que consiste a tarefa em causa;
 - Os **objetivos** caracterizam as metas a serem atingidas com a sua realização;

- O **local de investigação** indica as coordenadas onde esta deve ser iniciada;
 - O **caso** a que esta está ligada é a informação que permite contextualizar a tarefa;
 - Uma tarefa só se considera corretamente representada caso contenha todos estes dados.
- 4.1. Segundo o requisito 4, um Agente possui um plano de atividades que constitui um planeamento das tarefas (detalhadas no requisito 3) a executar durante uma sessão de trabalho;
- 4.2. Para esse efeito, a criação de um plano de atividades consiste numa associação de tarefas a um Agente que, ao serem importadas pelo *frontOffice*, serão agregadas num conjunto de maneira a formar a sessão de trabalho pretendida;
- 5.1. Pode ser constatado através do requisito 5 que os casos aceites (e consequentemente inseridos no sistema) pelo Diretor para investigação devem ser entregues a um Inspetor-Chefe;
- 5.2. De forma a concretizar esta associação, o *backOffice* deve permitir atribuir a responsabilidade de um dado caso (mencionado no requisito 2) ao perfil de um Inspetor-Chefe.
- 6.1. O sistema do *backOffice* deve possibilitar ao Diretor a criação de perfis de acesso correspondentes às permissões de um Inspetor-Chefe;
- 7.1. Deve ser permitido ao Diretor que, no *backOffice*, possa criar perfis para os seus Agentes acederem ao sistema. Para além disso, deve poder associá-los a uma determinada equipa chefiada por um Inspetor-Chefe (mencionado no requisito 8);
- 7.2. Esta associação vai fazer com que os Agentes em causa integrem apenas a equipa desse Inspetor-Chefe e, como tal, esta ligação deve ser única pois um Agente não deverá fazer parte de mais do que uma equipa de investigação simultaneamente (explicado no requisito 13).
- 8.1. O *backOffice* deve permitir ao Inspetor-Chefe ver os dados (código de identificação, nome próprio e apelido) de todos os Agentes que coordena;
- 9.1. Tal como mencionado no ponto 5, os Inspetores-Chefe são responsáveis por um conjunto de casos. Assim sendo deve ser permitido ao Inspetor-Chefe poder visualizar todos os seus casos bem como o estado em que se encontram;
- 9.2. Um Inspetor-Chefe, com base nos casos que coordena (ponto 9.1) cria e atribui tarefas aos Agentes que coordena. Estas tarefas constituem um plano de atividades, como mencionado no ponto 4.

- 9.3. Para além da atribuição de tarefas aos seus Agentes, um Inspetor-Chefe, sempre que necessário, deve poder verificar as tarefas que estão atribuídas a um dado Agente, bem como o seu estado de realização (terminadas / em curso);
 - 9.4. Caso um Inspetor-Chefe necessite remover uma tarefa que ainda não foi sincronizada pelo Agente, deve ser possível fazê-lo;
 - 9.5. Um Inspetor-Chefe só pode visualizar e gerir os casos a ele entregues;
 - 9.6. Uma tarefa não pode ser removida se já tiver sido sincronizada pelo Agente ao qual foi atribuída;
-
- 10.1. Sempre que um caso é resolvido, o Inspetor-Chefe responsável deve marcá-lo como concluído;
 - 10.2. Um caso não pode ser marcado como concluído se ainda existirem tarefas em curso relacionadas com o caso;
-
- 11.1. Além das funcionalidades específicas de um Inspetor-Chefe, este também pode atuar como um Agente, ou seja, tem acesso a todas as funcionalidades de um Agente;
 - 11.2. Como um Inspetor-Chefe é também um Agente (ponto 11.1), este tem a possibilidade de atribuir um plano de atividade a ele próprio;
-
- 12.1. O *backOffice* deve dar a possibilidade a um Inspetor-Chefe de recolher e analisar toda a informação sobre um dado caso e criar um *PDF* com essa informação. Este relatório são as considerações finais, mencionadas no ponto 2;
 - 12.2. Após a geração do *PDF* sobre um certo caso (ponto 12.1), o Inspetor-Chefe deve ter a possibilidade de o poder enviar por email a qualquer endereço eletrónico;
-
- 13.1. Agente deve poder importar o plano de atividade, a ele definido, para o seu *frontOffice*;
 - 13.2. Um Agente apenas recebe planos de Atividade do Inspetor-Chefe que o coordena; Agente deve poder consultar, no seu *frontOffice*, o seu plano de atividades a fim de o poder executar;
-
- 14.1. Um Agente apenas pode visualizar os casos que estejam ao encargo do seu Inspetor-Chefe;
-
- 15.1. Um Agente deve poder realizar e cumprir cada tarefa do plano de atividades;
-
- 16.1. Existe a possibilidade do Agente suspender uma tarefa em execução.
 - 16.2. Sempre que um Agente seleciona uma tarefa, deve existir a possibilidade do *frontOffice* apresentar o percurso, indicando o caminho a percorrer para alcançar o

último local assinalado (caso este tenha sido indicado sob a forma de dados recolhidos, dados estes mencionados no ponto 17.1). Caso não exista nenhum registo, é utilizado o local de investigação da tarefa;

- 17.1. Quando um Agente está a realizar uma tarefa, o *frontOffice* deve permitir tirar fotografias, notas escritas e de voz. Todas estas ações devem ser registadas e guardadas (na base de dados do *frontOffice*) juntamente com a localização do sítio onde foram realizadas. Além disso, estas informações ficam associadas à tarefa em questão;
- 18.1. Além das funcionalidades mencionadas no ponto 17.1, um Agente deve poder obter informações complementares (através do *frontOffice*) sobre um determinado assunto. Essa obtenção será realizada com o auxílio de um motor de busca;
- 19.1. Quando o Agente se conecta ao *backOffice*, todas as tarefas que realizou durante a sua sessão de trabalho, bem como os dados recolhidos (mencionados no requisito 17) relativos a cada uma das tarefas que concretizou, devem ser exportados para o sistema central;
- 19.2. Apenas as tarefas que estão dadas como concluídas no *frontOffice* do Agente devem ser sincronizadas (exportadas) com o *backOffice*;
- 19.3. As tarefas que esperam a sua concretização devem permanecer no *frontOffice*.
- 20.1. O *backOffice* deve analisar as gravações de voz registadas pelos Agentes, traduzindo-as para um documento anotado;
- 20.2. Esta análise é feita com base em *tags* que têm o objetivo de indicar que informação se procede à sua ocorrência;
- 20.3. Quando a análise do registo de voz deteta uma *tag*, dependendo do significado que lhe é atribuído, a informação correspondente é processada de forma adequada.
- 21.1. Toda a informação relativa aos casos e às entidades da empresa devem estar armazenadas numa base de dados relacional presente no *backOffice*;
- 21.2. Relativamente aos dados das tarefas recolhidos durante o trabalho de campo dos Agentes, estes encontram-se armazenados na base de dados do dispositivo móvel enquanto não são exportados para o *backOffice*;
- 21.3. Depois dos dados recolhidos nas investigações serem exportados para o *backOffice*, estes deixam de estar presentes na base de dados do *frontOffice*.

2.2.2. Não Funcionais

22. O dispositivo móvel onde o *frontOffice* será instalado deve estar munido de uma câmara fotográfica, gravador de voz, acesso à Internet e GPS.

3. Modelos de Sistema

3.1. Diagramas UML

3.1.1. Diagramas de Use Case

De maneira a melhor representar e diferenciar a interação com o sistema global, foram desenvolvidos dois modelos de *Use Cases* distintos, um para o Front-Office e outro para o Back-Office. Cada um destes reflete a interação dos vários tipos de utilizadores com o respetivo subsistema, sendo que por vezes é também representada a interação entre os dois subsistemas.

3.1.1.1. Diagrama de Use Case - Back-Office

Neste diagrama é apresentada a interação entre os vários tipos de utilizadores com o subsistema de Back-Office. Todos os tipos de utilizadores da empresa poderão interagir com este subsistema. Contudo, estes realizam ações diferentes no sistema, o que leva a que o sistema forneça certas funcionalidades apenas a um tipo de utilizadores. Conforme especificado na secção de requisitos, as diversas funcionalidades que o Back-Office deve fornecer a cada tipo de interveniente estão ilustradas no diagrama seguinte:

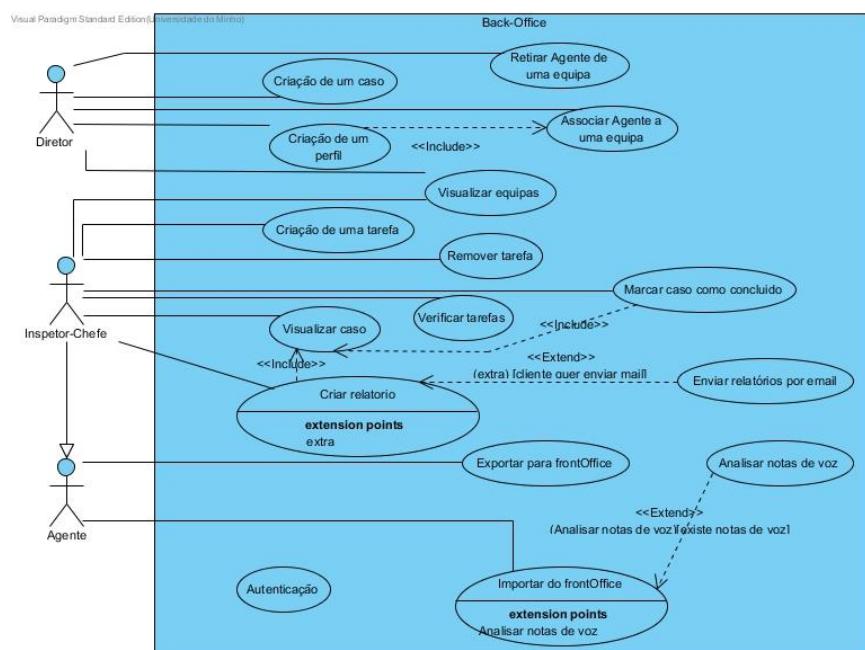


Figura 3 Diagrama de Use Cases - BackOffice

Analisando o diagrama, pode-se verificar que está representada, através de uma seta, uma relação de generalização entre atores, que envolvem os atores Inspetor-Chefe e Agente. Esta generalização efetuada reflete o requisito 11.1. presente na secção 2.2.1. Como consequência desta generalização, fica implícito que qualquer funcionalidade apresentada no diagrama anterior fornecida ao Agente pode ser executada também por um Inspetor-Chefe.

Cada Use Case/funcionalidade presente no diagrama anterior foi devidamente especificado, podendo essa informação ser consultada na opção de abrir detalhe de Use Case, no separador de descrição. Esta descrição foi efetuada em formato tabular, em que se descreve sucintamente o Use Case, as pós e pré-condições, bem como o processo de interação entre o ator e o subsistema. De forma exemplificar o processo, serão de seguida apresentados alguns exemplos desta especificação.

3.1.1.1. Exemplo de especificação de Use Case - Criação de uma tarefa

Na seguinte imagem é possível observar a descrição em formato tabular do Use Case “Criação de uma tarefa”, que tem como ator um Inspetor-Chefe. Na linha de breve descrição é descrito que esta funcionalidade permite criar uma tarefa e atribui-la a um Agente de campo. É indicada como pré-condição que existem agentes a quem atribuir a tarefa, e como pós-condição que a tarefa é criada e atribuída com sucesso.

De seguida é descrito o comportamento normal, em que se descreve como se desencadeia o processo, de forma sequencial, e de que forma o ator e o sistema atuam perante ações efetuadas por eles. Inicialmente, o ator solicita o serviço, e o sistema procura casos e agentes da equipa do Inspetor e responde, pedindo a inserção da informação caracterizadora de uma tarefa (conforme especificada no requisito funcional de sistema 3.1 da secção 2.2.1). Após este pedido, o Inspetor vai inserir e submeter a informação. Caso a informação inserida esteja devidamente fornecida, o sistema vai criar por fim a tarefa, associa-la à informação inserida informar o sucesso do processo. Caso contrário, é transmitido ao Inspetor que a informação não está correta e é pedida a reinserção da mesma.

● **Criação de uma tarefa**

The screenshot shows a software interface for managing use cases. At the top, there's a menu bar with options like Info, Use Case Notes, Flow of Events, Details, Requirements, Diagrams, Test Plan, References, and Description. Below the menu is a toolbar with various icons. The main area is titled 'Main' and contains a table for the 'Super Use Case'. The table includes fields for Author (Grupo 3), Date (25/Abr/2016 18:57:50), Brief Description (Permite a criação de uma tarefa e atribuição da mesma a um agente), Preconditions (Existem agentes para atribuir tarefas;), and Post-conditions (Tarefa criada e atribuída com sucesso;). A section titled 'Flow of Events' contains a table with steps 1 through 7. Step 1: Actor Input 'Solicita criação de tarefa' and System Response 'Procura Casos e Agentes da sua equipa'. Step 2: Actor Input '' and System Response 'Solicita informação caracterizadora da tarefa (local de investigação, objetivo, descrição, título, agente e caso)'. Step 3: Actor Input '' and System Response 'Valida informação'. Step 4: Actor Input 'Insere e submete informação' and System Response 'Cria a tarefa c/ informação associada'. Step 5: Actor Input '' and System Response 'Informa sucesso na criação e atribuição'. Step 6: Actor Input '' and System Response ''. Step 7: Actor Input '' and System Response ''. An 'Alternativa 1' section is also present, with steps 1 and 2. Step 1: Actor Input '' and System Response 'Informa erro nos campos'. Step 2: Actor Input '' and System Response 'Volta a 2'.

Figura 4 Especificação do Use Case "Criação de uma tarefa"

3.1.1.1.2. Exemplo de especificação de Use Case - Exportar para Front-Office

Neste exemplo é descrito o processo de exportação de tarefas para o Front-Office. É assumido que existe informação para ser exportada, que existe conexão entre os dois subsistemas e que após o processo esta informação foi transferida com sucesso.

Na descrição do comportamento normal do sistema (Flow of Events) é descrito a interação com o sistema. Inicialmente, o Front-Office solicita a exportação ao Back-Office, e envia as suas credenciais. O Back-Office valida as suas credenciais, e caso sejam válidas, verifica quais as tarefas por sincronizar e envia-as. Após o envio, assinala internamente as tarefas como sincronizadas.

Além do comportamento normal, foi também contemplada uma exceção, que caso ocorra um erro nas credenciais enviadas pelo Front-Office, o Back-Office informa que ocorreu um erro nas credenciais, o que implica que a informação não foi transferida.

Exportar para frontOffice		
Info	Use Case Notes	Flow of Events
Details	Requirements	Diagrams
Test Plan	References	Description
	Agency FB	8
Main		
Super Use Case		
Author	Grupo 3	
Date	25/Abr/2016 21:43:13	
Brief Description	A informação por sincronizar é transferida para o front-office; Existe conexão;	
Preconditions	Há informação para sincronizar;	
Post-conditions	A informação é exportada com sucesso	
Flow of Events	Front-Office Input	System Response
1	Solicita exportação	
2	Envia credenciais	
3		Valida credenciais
4		Verifica quais as tarefas a serem enviadas
5		Envia tarefas
6		Marca as tarefas como sincronizadas
Excepção 1 [Credenciais inválidas] (Passo 2)	Actor Input	System Response
1		Informa erro nas credenciais

Figura 5 Especificação do Use Case "Exportar para o frontOffice"

3.1.1.2. Diagrama de Use Case - Front-Office

O diagrama de Use Case referente ao subsistema FrontOffice, tem os seguintes atores: Inspetor-Chefe e Agente. Uma vez que foi referido que tudo o que um Agente pode realizar, também um Impetor-Chefe o pode (referido no requisito 11.1. na secção de Requisitos de Sistema), estes atores foram sujeitos a uma generalização. Ou seja, todos os *use case* de Agente, também são *use cases* de um dado Inspetor-Chefe.

Foram definidos vários *use cases* que representam as diversas funcionalidades dos atores no sistema FrontOffice, bem como funcionalidades do sistema. Esta definição de *use cases* teve como suporte os requisitos anteriormente mencionados numa secção anterior (secção 2.2. Requisitos de Sistema). A título de exemplo deste processo, vamos proceder à explicação do *use case* Realizar tarefa. Uma das principais funcionalidades que o FrontOffice oferece é a de permitir ao seu utilizador realizar uma dada tarefa que tenha associada, com todas as opções subjacentes. Esta funcionalidade é especificada no requisito 15.1. da secção Requisitos de Sistema. Quando de uma realização de tarefa, é apresentada ao utilizador o seu plano de atividades (requisito 13.2.), daí se ter criado um *use case* Visualizar plano de atividades e se ter associado ao *use case* Realizar tarefa através de um <>include<>. Posteriormente, a fim de poder realizar a tarefa, o utilizador tem a possibilidades de tirar fotografias, notas escritas e

notas de voz, tal como referido no requisito 17.1. Desta forma, foram criados os seguintes *use cases*: Tirar fotografia, Tirar nota de voz e Tirar nota escrita. O sistema também permite ao utilizador obter informações complementares sempre que achar necessário (tal como referido no requisito 18.1.), bem como obter o percurso para um determinado local, neste caso tem duas possibilidades: caminho para o ultimo local assinalado, ou para o local de inicio da atividade. Esta última funcionalidade está referida no requisito 16.2., e com base nestas duas últimas funcionalidades foram criados os *use cases* Obter inf. complementares e Apresentar caminho. A associação destes *use cases* com o *use case* que se está a exemplificar foi também através de `<<include>>`.

Da análise dos requisitos é verificado que aquando de uma obtenção de uma fotografia, de uma nota de voz, bem como de uma nota escrita, é necessário registar a localização do acontecimento (requisito 17.1.). Assim sendo, foi criado e associado um *use case* denominado de Registrar localização.

O diagrama de *use cases* relativo ao subsistema *FrontOffice*, contendo os atores identificados bem como todos os *use cases* relacionados são representados na seguinte figura.

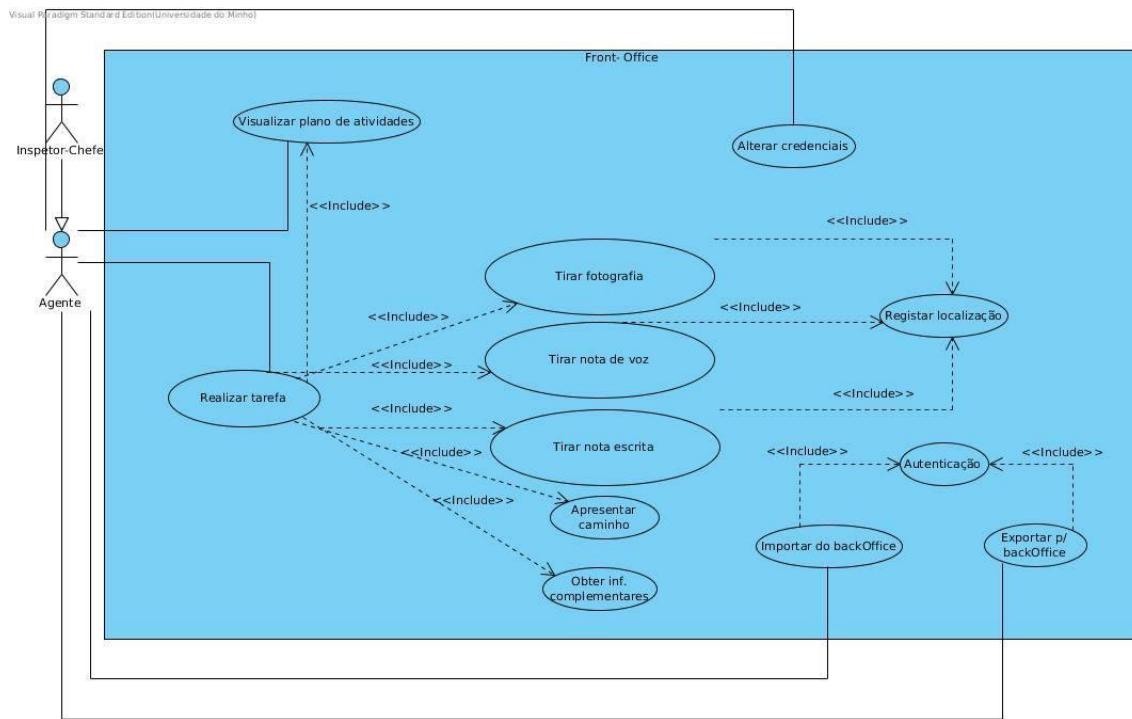


Figura 6 Diagrama de Use Case - FrontOffice

3.1.1.2.1. Exemplo de especificação de Use Case – Realizar tarefa

O passo seguinte à construção do diagrama de *use cases* passa pela especificação dos mesmos. De forma a exemplificar este processo, será demonstrado a especificação do *use case* Realizar tarefa.

O processo de especificação inicia-se pela realização de uma breve descrição do *use case*, indicação da pré-condição e da pós-condição. Na descrição é referido que o *use case* em questão passa pela realização de uma dada tarefa existente no plano de atividades, sendo que no final, a tarefa é dada como concluída ou suspensa. A pré-condição indica que tem de existir tarefas por concluir no plano de atividades, a pós-condição indica que a tarefa que se pretende realizar tem de ser dada como concluída ou suspensa. Se a pré-condição e a pós-condição se verificarem no final, a sequência de eventos é denominada de normal.

De seguida, é especificado de forma tabular as diversas interações entre o ator (neste caso o ator é um determinado Agente ou Inspetor-Chefe) e o sistema. Inicialmente é mostrado ao utilizador o seu plano de atividades (foi feito um *include* de Visualizar plano de atividades), ou seja, é referenciado outro *use case*. Este *use case* é bastante simples, consiste apenas numa solicitação do ator em visualizar o seu plano de atividades, por sua vez, o sistema procura as atividades associadas ao ator e mostra-as. De seguida, de volta ao *use case* Realizar tarefa, o ator escolhe uma determinada tarefa que pretende realizar, o sistema procura-a e apresenta ao ator as várias ações que o ator pode fazer para realizar a tarefa. As opções são as seguintes: marcar a tarefa como concluída, marcar a tarefa como suspensa (estas duas representam terminar o *use case* com um comportamento normal (fluxo principal)), tirar uma fotografia, tirar uma nota de voz, tirar um nota escrita, obter informações complementares e obter percurso para um local. Desta modo, foram implementadas diversas alternativas de forma a permitir ao ator escolher uma opção. Por exemplo, se a opção for de tirar uma fotografia, passa para a Alternativa 2 que tem uma referência ao *use case* Tirar fotografia. Este *use case* inicia-se pela permissão do sistema ao ator de tirar a fotografia, o ator tira a fotografia e de seguida será registada a localização do local onde a fotografia é tirada. Para registar a localização, é referenciado o *use case* Registrar localização que consiste apenas no sistema em determinar a localização e associar à fotografia. Este processo de registo de localização está presente também quando se obtém uma nota de voz ou uma nota escrita.

Aquando da escolha de uma outra qualquer opção que não a de tirar fotografia, o processo segue o mesmo raciocínio, passa para a alternativa relativa à sua opção e faz o pretendido. De realçar que no fim destas alternativas volta-se ao passo no qual se escolhe uma opção, permitindo realizar diversas ações aquando da realização de uma dada tarefa. De notar que quando a opção escolhida é a de marcar a tarefa como concluída, não existe passagem para uma alternativa, mas sim, o fluxo segue no fluxo principal e o sistema marca a tarefa como concluída e a pré-condição é satisfeita. A outra forma de satisfazer a condição, tal como

anteriormente referido, é marcar a tarefa como suspensa. Nesta especificação, este comportamento é representado com uma alternativa, isto é, aquando da escolha da opção marcar a tarefa como suspensa, passa para a alternativa correspondente e o sistema marca a tarefa como suspensa e o fluxo é terminado com a pré-condição satisfeita.

De seguida é apresentado a especificação do *use case Realizar tarefa* que acabou de ser explicado:

Super Use Case		
Author	grupo 3	
Date	25/Abr/2016 17:34:26	
Brief Description	O agente realiza uma dada tarefa do plano de atividades, concuindo-a ou não	
Preconditions	Existem tarefas por concluir no plano de atividades	
Post-conditions	Tarefa dada como concluída ou suspensa	
Flow of Events	Actor Input	System Response
	1	<<include>> Visualizar plano de atividades
	2 Seleciona tarefa	
	3	Pesquisa tarefa selecionada
	4	Apresenta opções
	5 Escolhe opção concluir tarefa	
	6	Regista tarefa como concluída
Alternativa 1 [Escolhe opção de indicação de caminho] (Passo 5)	Actor Input	System Response
	1 Escolhe opção de indicação de caminho	
	2	<<include>> Apresentar caminho
	3	Volta ao passo 4
Alternativa 2 [Escolheu tirar fotografias] (Passo 5)	Actor Input	System Response
	1 Escolhe opção tirar fotografias	
	2	<<include>> Tirar fotografia
	3	Volta ao passo 4
Alternativa 3 [Escolheu tirar nota de voz] (Passo 5)	Actor Input	System Response
	1 Escolheu opção tirar nota de voz	
	2	<<include>> Tirar nota de voz
	3	Volta ao passo 4
Alternativa 4 [Escolheu tirar nota escrita] (Passo 5)	Actor Input	System Response
	1 Escolheu opção tirar nota escrita	
	2	<<include>> Tirar nota escrita
	3	Volta ao passo 4
Alternativa 5 [Escolheu obter informações complementares] (Passo 5)	Actor Input	System Response
	1 Escolheu opção obter informações complementares	
	2	<<include>> Obter info complementares
	3	Volta ao passo 4
Alternativa 6 [Escolheu suspender tarefa] (passo 5)	Actor Input	System Response
	1 Escolheu opção de suspender tarefa	
	2	Regista tarefa como suspensa

Figura 7 Especificação do Use Case "Realizar tarefa"

3.1.2. Diagramas de sequência

De forma a complementar a informação descrita e detalhada sobre as funcionalidades fornecidas através dos diagramas de Use Cases, foram desenvolvidos diagramas de sequência. Com este outro tipo de diagramas, é possível obter uma melhor visão sobre a arquitetura, havendo maior foco no ordenamento temporal das trocas de mensagens entre os intervenientes. Este tipo de modelo está mais próximo do código que irá ser desenvolvido e facilita a transição para essa mesma fase de desenvolvimento, tornando a tarefa de compreensão do pretendido pela equipa de desenvolvimento bastante mais facilitada.

A cada Use Case presente nos diagramas anteriores fez-se corresponder um diagrama de sequência, traduzindo estes a interação entre atores e sistema conforme foi detalhado em formato tabular em cada Use Case.

Com vista a melhor descrever e facilitar este processo de desenvolvimento dos diagramas de sequência, serão apresentados de seguida alguns exemplos dos diagramas desenvolvidos.

3.1.2.1. Exemplo de Diagrama de Sequência- Criação de uma tarefa

De forma a completar o exemplo anteriormente fornecido para a descrição do Use Case “Criação de uma tarefa”, é agora apresentado e descrito o correspondente diagrama de sequência. Neste diagrama é possível visualizar o Inspetor-Chefe, o ator que desencadeia e usufrui desta funcionalidade e o sistema com que interage, o Back-Office assim como também as respetivas linhas de tempo destes intervenientes. O processo inicia-se com a comunicação do Inspetor ao sistema que pretende criar uma tarefa. Este reage, procurando internamente os casos e agentes coordenados pelo Inspetor. De seguida verifica-se um ciclo em que enquanto não for inserida corretamente a informação relativa à tarefa, solicitada pelo sistema, é informado o erro nessa informação e não é permitido prosseguir sem que a informação seja válida. Quando é validada a informação, o sistema associa a tarefa ao Agente, informando depois o Inspetor que o processo terminou com sucesso.

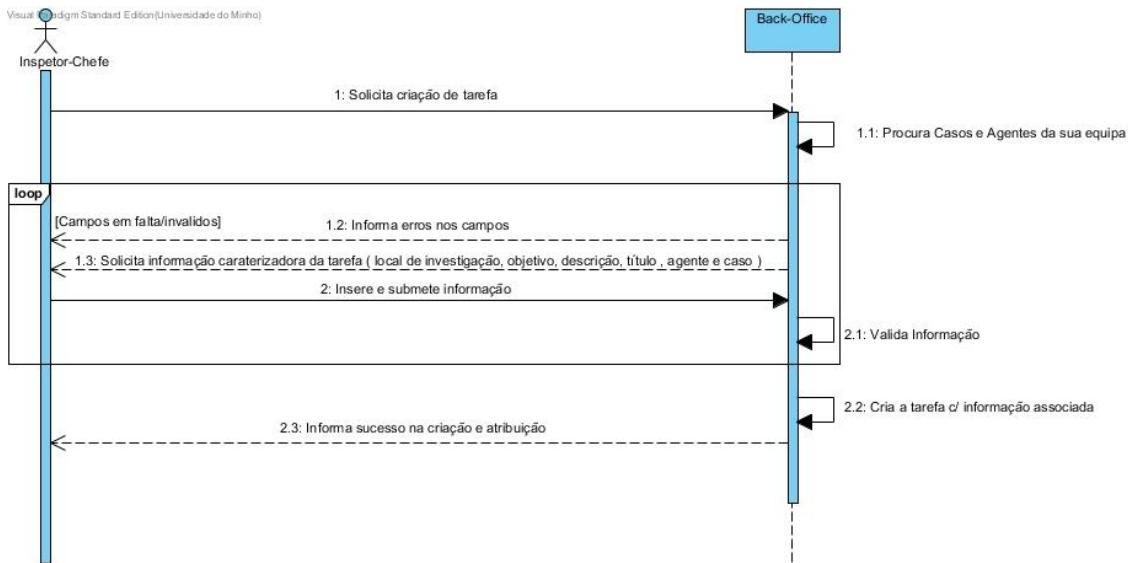


Figura 8 Diagrama de Sequência "Criação de uma tarefa"

3.1.2.2 Exemplo de Diagrama de Sequência- Exportar para Front-Office

Dando sequência ao exemplo de Use Case apresentado anteriormente para esta funcionalidade, apresenta-se de seguida um exemplo de um diagrama de sequência que reflete o comportamento descrito anteriormente.

Neste exemplo é visível um refinamento ao nível da representação do sistema. Na descrição apresentada anteriormente em formato tabular, na especificação do Use Case, era possível observar que o ator não interagia com o sistema, mas sim o subsistema Front-Office. Como se efetuou a divisão em dois diagramas de Use Case para melhor representar a interação com cada subsistema, poderia ser inferido que o sistema com que se interage neste Use Case seria apenas o Back-Office. Contudo é possível observar neste novo tipo de diagrama que na realidade esta funcionalidade disponibilizada na realidade envolve os dois subsistemas.

O processo inicia-se com o Front-Office a solicitar a exportação ao Back-Office, e a enviar depois as suas credenciais. De seguida o Back-Office valida as credenciais (podendo aqui ocorrer um erro nas credenciais, que será detetado caso se verifique e que será reportado ao utilizador, o que leva a que o processo termine), verifica internamente quais as tarefas a ser enviadas, envia as tarefas para o Front-Office, marcando-as de seguida como sincronizadas.

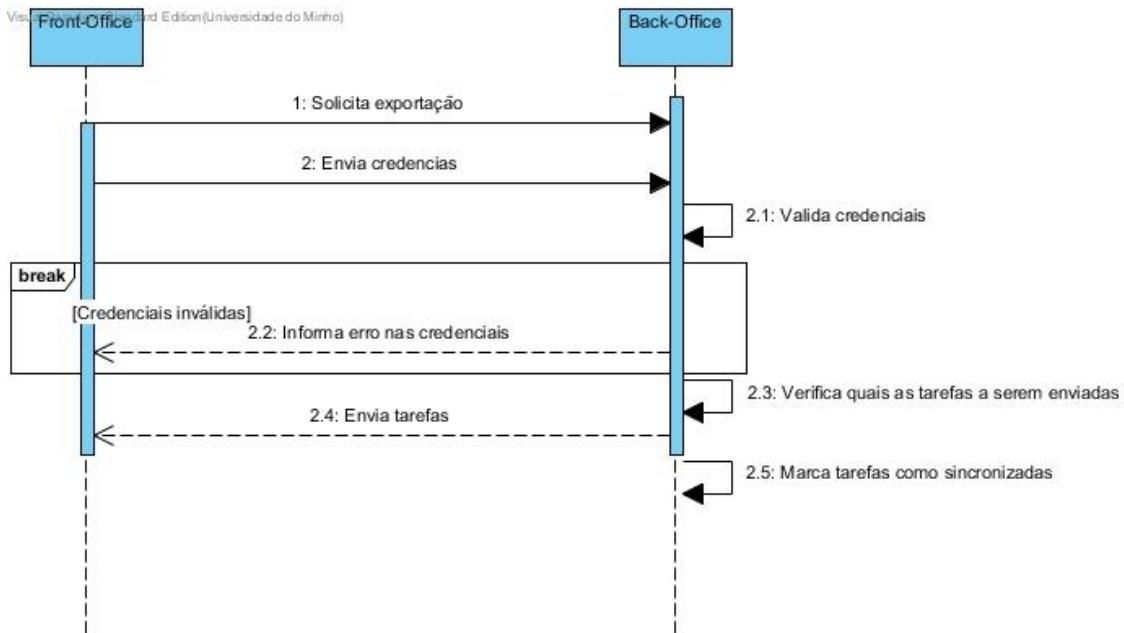


Figura 9 Diagrama de Sequência "Exportar para o frontOffice"

3.1.2.3. Exemplo de Diagrama de Sequência- Realizar tarefa

De forma a exemplificar um diagrama de sequência relativo a um determinado *use case* e para dar seguimento ao exemplo escolhido para explicar uma especificação, vai se proceder à exemplificação do diagrama de sequência relativo ao *use case Realizar tarefa*.

Inicialmente, uma vez que é mostrada ao ator o seu plano de atividades, e na especificação do *use case* existe um <<include>>Visualizar plano de atividades, é feita uma referência a outro diagrama (através do operador *ref*). A figura seguinte é referente ao diagrama de sequência DSS – Visualizar plano de atividades que acabou de ser mencionado. Neste diagrama, o Agente manifesta intenção de visualizar plano de atividades ao sistema através de uma mensagem, por sua vez, o FrontOffice procura as atividades no próprio FrontOffice e é representado com uma *self message*. Finalmente o FrontOffice responde ao Agente fornecendo-lhe as atividades.

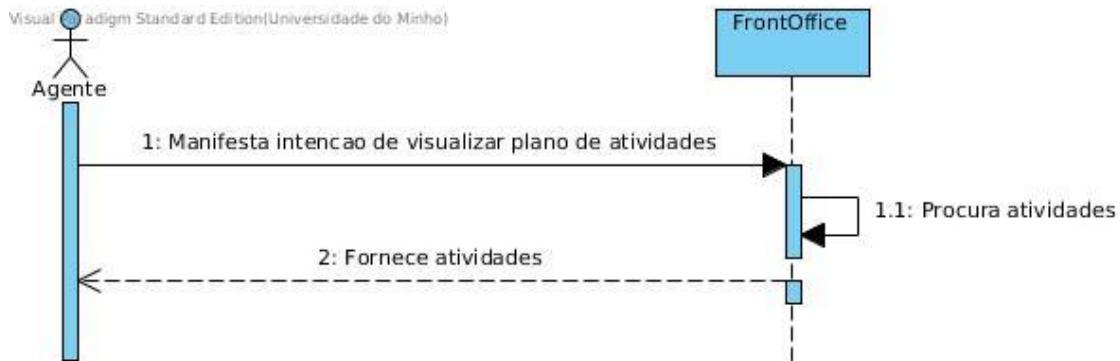


Figura 10 Diagrama de Sequência "Visualizar plano de atividades"

De volta ao diagrama de sequência DSS – Realizar tarefa, o processo continua enviando uma mensagem do ator Agente para o FrontOffice no qual indica a tarefa que pretende realizar, sendo que o subsistema FrontOffice pesquisa a tarefa selecionada no sistema, e esta pesquisa do sistema no próprio sistema é representada através de uma *self message*. Após este processo, é criado um *loop* que permite ao ator realizar várias vezes uma das ações permitidas enquanto a tarefa não seja marcada como concluída ou suspensa. A cada iteração do *loop* é apresentado as opções do sistema ao utilizador, isto é representado com uma mensagem denominada de resposta. Logo após, existe um operador *alt* que define fragmentos alternativos, neste caso, permite dividir em varias opções de escolha por parte do ator, uma vez que os fluxos possíveis são mutuamente exclusivos, pelo que apenas um deles será seguido. Caso a opção escolhida seja marcar a tarefa como concluída ou marcar a tarefa como suspensa, entra no operando que contem a condição respetiva e através de uma mensagem do Agente para o FrontOffice indica que escolheu marcar a tarefa como suspensa ou concluída e o sistema, internamente, vai registar a tarefa como suspensa ou concluída. O fluxo termina aqui (para estas situações) visto que saindo do *alt*, vai ser verificada a condição do *loop* e como a condição não é verificada não entra novamente no *loop* e termina aqui o fluxo, sendo a pós-condição satisfeita tal como é pretendido. Caso a escolha da opção não seja nenhuma das duas mencionadas anteriormente, ao entrar no operando correspondente vai fazer referência a outro diagrama de sequência (por exemplo, se a opção for a de tirar fotografia, vai fazer referência ao Diagrama de Sequência DSS – Tirar fotografia) e depois, ao sair do *alt* e verificar a condição do *loop*, vai ver que a condição é satisfeita, permitindo desta forma voltar a poder escolher uma opção.

Na seguinte figura está representado o diagrama de sequência que se exemplificou, nomeadamente: DSS – Realizar tarefa:

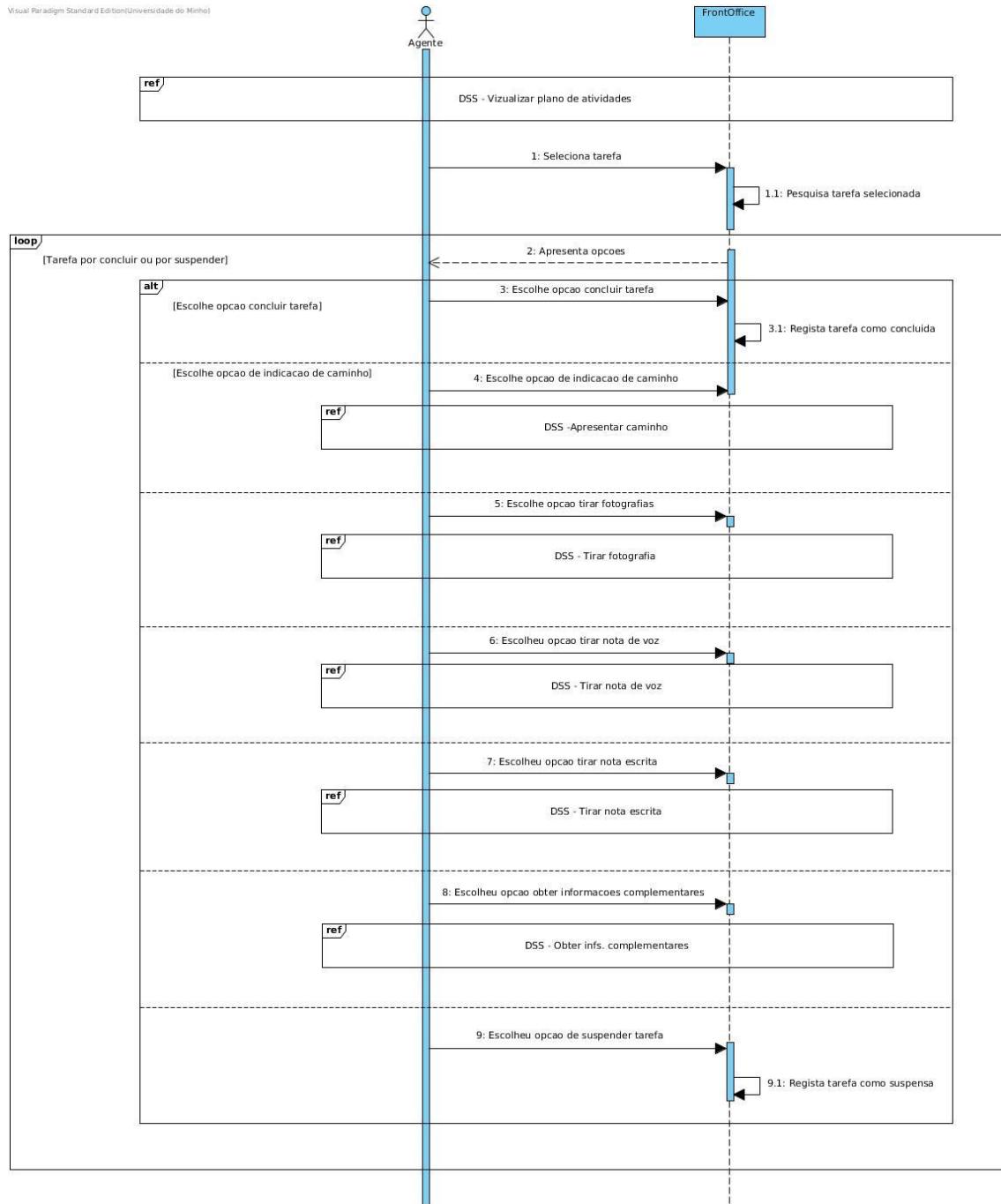


Figura 11 Diagrama de Sequência "Realizar tarefa"

3.1.3 Diagramas de Atividade

Os diagramas de atividade podem ter uma grande importância na especificação de sistemas. A sua utilidade no contexto deste projeto é relevante, visto que ajudam a especificar o comportamento dos elementos de software a desenvolver, o fluxo de dados que as várias funcionalidades geram, bem como o encadeamento dos processos envolvidos.

Como tal, de forma a complementar a informação já transmitida através dos outros tipos de modelos apresentados anteriormente, procedeu-se ao desenvolvimento de um diagrama de atividade para cada funcionalidade que o sistema suportará. Desta forma facilitar-se-á a tarefa da equipa de desenvolvimento da peça de software, de forma a fornecer-lhe uma espécie de guião a seguir que levará o programa a corresponder ao pedido pelo cliente, conforme foi especificado na secção de análise de requisitos.

De forma a melhor descrever o processo de desenvolvimento deste tipo de diagramas, e ajudar a traduzir a informação neles contida, apresentam-se de seguida alguns exemplos desenvolvidos deste tipo de diagramas.

3.1.3.1. Exemplo de diagrama de atividade – Criação de uma tarefa

Neste diagrama é possível observar a interação de um ator, um Inspetor-Chefe, com o subsistema de Back-Office. No diagrama é visível uma separação entre estes dois intervenientes, de forma a separar quais as ações efetuadas por cada um deles.

O processo inicia-se por iniciativa do Inspetor, conforme é indicado pelo círculo preto presente no seu separador. O Inspetor começa por solicitar a criação de uma tarefa, cabendo de seguida ao Back-Office procurar os casos e agentes da equipa coordenada pelo Inspetor e pedir a informação caracterizadora da tarefa a ser criada. Após este pedido, o Inspetor deve inserir a informação e submete-la. O Back-Office vai validar a informação previamente inserida e vai decidir a ação a tomar conforme a validade dos dados inseridos. Irá voltar a pedir a sua reinserção por parte do Inspetor, caso não seja válida, ou prosseguirá. Se prosseguir, vai criar a tarefa, e atribuí-la à informação escolhida. Finalmente, o subsistema vai informar o sucesso do processo e dá por terminada a execução da funcionalidade.

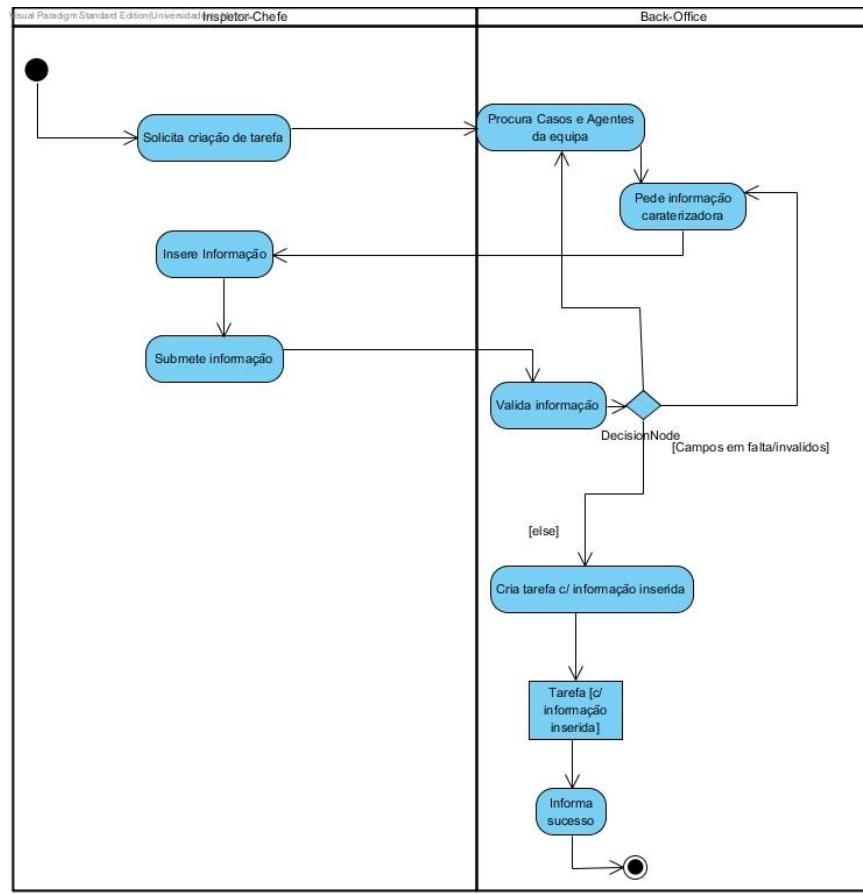


Figura 12 Diagrama de Atividade "Criação de uma tarefa"

Nas ações mais importantes e que suscitem mais dúvidas, é disponibilizada uma descrição da secção de especificação de cada ação. Nessa secção procedeu-se a uma descrição mais detalhada da ação, de forma a impedir várias interpretações e esclarecer em que consiste em concreto cada ação.

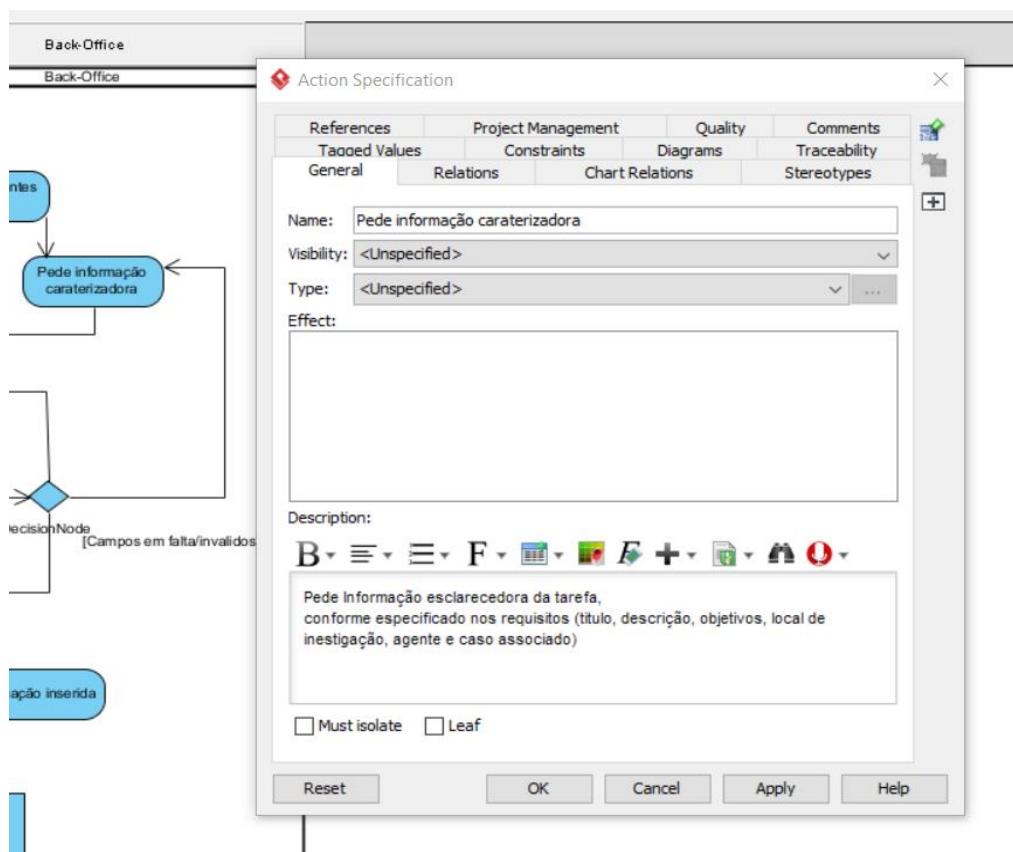


Figura 13 Descrição da ação "Pede informação caraterizadora"

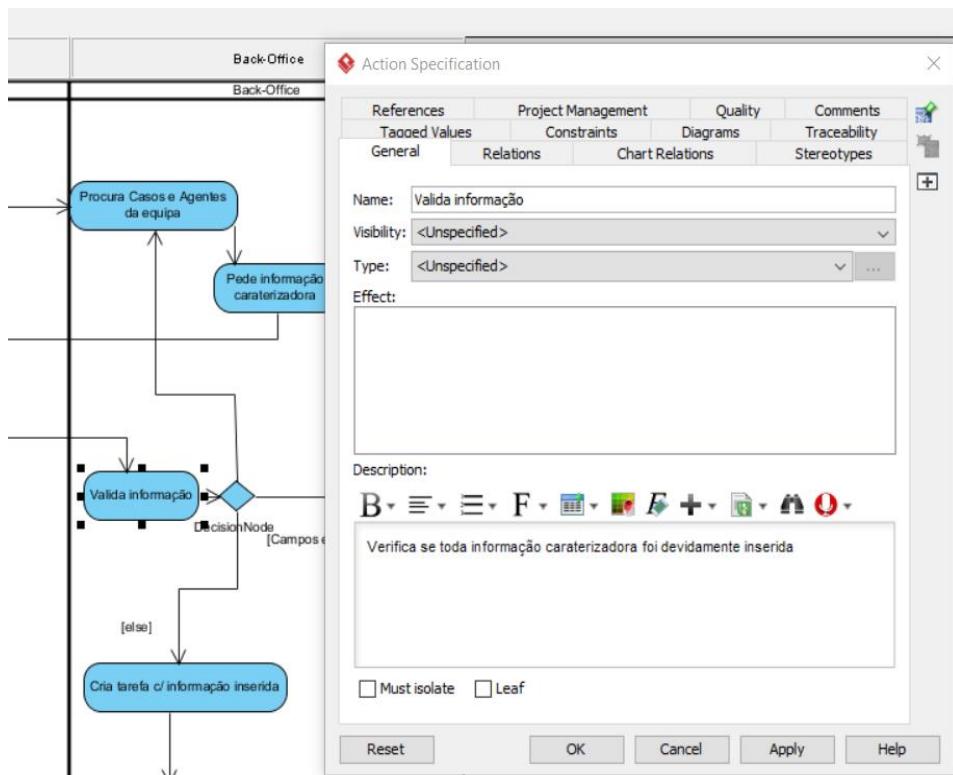


Figura 14 Descrição da ação "Valida informação"

3.1.3.2. Exemplo de diagrama de atividade – Exportar para Front-Office

Neste diagrama é possível observar uma separação entre os 2 tipos de intervenientes desta funcionalidade – Back-Office e Front-Office. O processo inicia-se com o Front-Office a solicitar a exportação e a enviar as credencias do agente a que pertence. Caso essas credenciais sejam válidas, o Back-Office continuará o processo normalmente. Caso contrario, o processo é interrompido, surgindo a informação que existe um erro nas credenciais. Seguindo o processo pela primeira alternativa, o Back-Office vai verificar quais as tarefas a ser enviadas, e enviá-las para o Front-Office. Segue-se uma bifurcação em que podemos ver como os dois subsistemas reagem diferentemente depois desta ultima ação. O Back-Office vai marcar as tarefas como sincronizadas (de onde resultam objetos “Tarefa” com estado “Sincronizada”) e terminar o processo. Já o Front-Office vai receber as tarefas (objetos “Tarefa” com estado “não sincronizada”), e armazená-las.

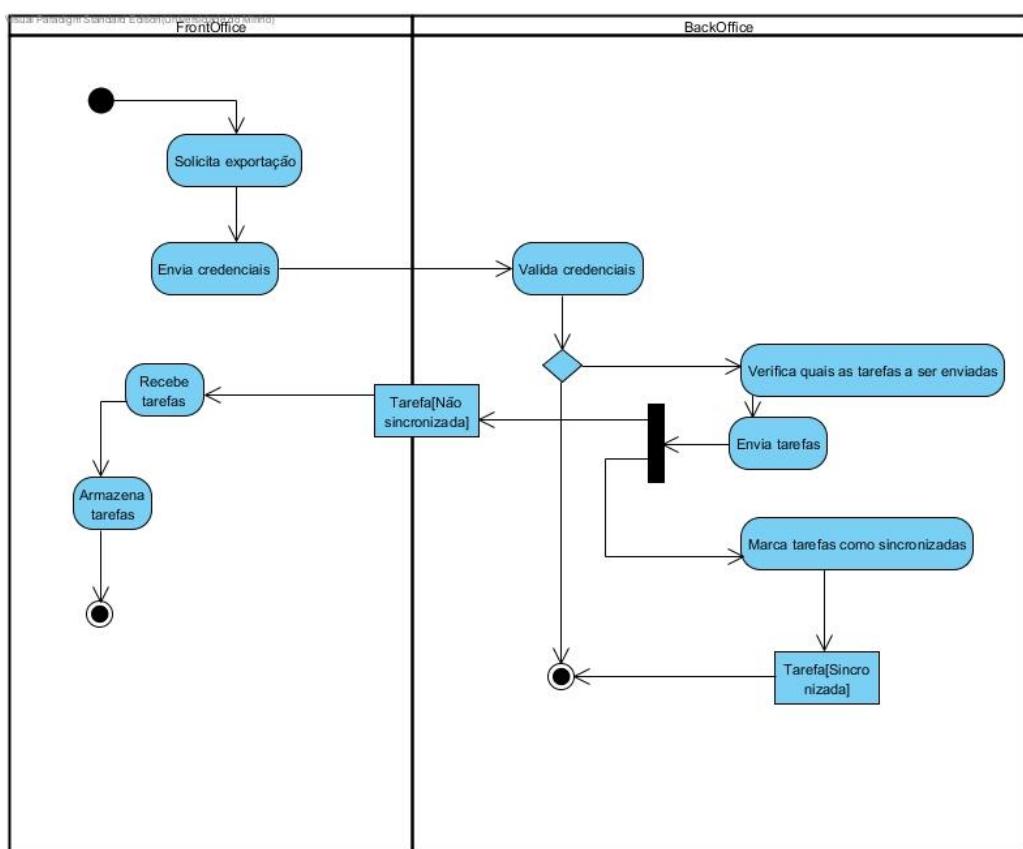


Figura 15 Diagrama de Atividade “Exportar para frontOffice”

Tal como referido no exemplo anterior, foi desenvolvida uma especificação para as ações mais importantes e que suscitam mais dúvidas envolvidas no processo. De seguida apresenta-se um desses exemplos.

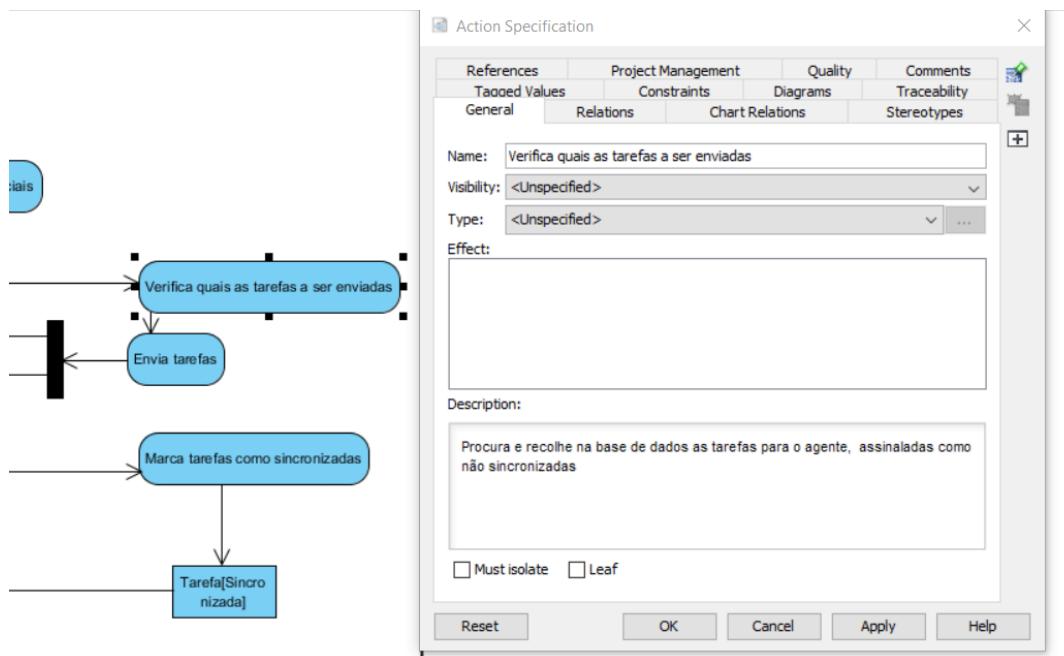


Figura 16 Descrição da ação "Verifica quais as tarefas a ser enviadas"

3.1.3.3. Exemplo de diagrama de atividade – Realizar tarefa

O diagrama de atividades que se segue representa o processo de realização de uma tarefa por parte de um Agente com auxílio do FrontOffice. Tendo em conta estes dois intervenientes, foram criadas partições, uma vez que partições permitem capacidade expressiva de associar papéis e responsabilidades às atividades.

O processo inicia-se por intenção do Agente, sendo, desta forma, representado o início do processo por um círculo preto na partição Agente. O Agente indica que pretende realizar uma tarefa e este acontecimento é representado com uma Ação, de seguida o FrontOffice procura as tarefas todas associadas ao Agente dando origem a um *object* que chamamos de Tarefas. Seguidamente, o Agente seleciona uma tarefa em específico, o FrontOffice pesquisa a tarefa dando origem a um *object* denominado por Tarefa. Neste *object* foi designado o seu estado, mais concretamente, o estado de Tarefa é que ainda está por concluir ou suspender. O passo seguinte é o de o Agente escolher a opção que pretende realizar (tirar fotografia, tirar nota de voz, tirar nota escrita, obter informações complementares, visualizar percurso para um dado local, concluir ou suspender a tarefa). Este passo é representado com um nodo de decisão e está presente na

partição Agente. Com base na escolha do Agente, o fluxo segue diferentes caminhos, por exemplo, se a escolha for de ver o caminho, passa para um novo nodo de decisão no qual tem de escolher qual dos caminhos pretende ter informação. Com base nesta opção de caminho, o FrontOffice pesquisa o caminho e é criado um object Caminho ou Ultimo caminho. De seguida o FrontOffice apresenta o caminho ao Agente e finalmente volta ao nodo de decisão presente na participação Agente que representa a decisão da opção a realizar.

Caso a opção escolhida seja a de tirar uma fotografia, uma nota de voz ou nota escrita, é criado um *object* relacionado com a opção e com um estado no qual indica que não existe localização associada. De seguida, no FrontOffice, vai ser pesquisada a localização atual, registando e associando ao *object* anteriormente criado, tendo desta forma o mesmo *object* com um novo estado (com localização associada). Finalmente, volta ao nodo de decisão que permite escolher qual das opções quer realizar.

Por sua vez, caso a opção escolhida seja a de concluir uma tarefa ou a de suspender uma tarefa, o *object* Tarefa muda de estado, passando para concluída ou suspensa e, finalmente, vai para um nodo de fim de atividade. Apenas escolhendo marcar a tarefa como concluída ou suspensa se consegue ir para o nodo de fim de atividade.

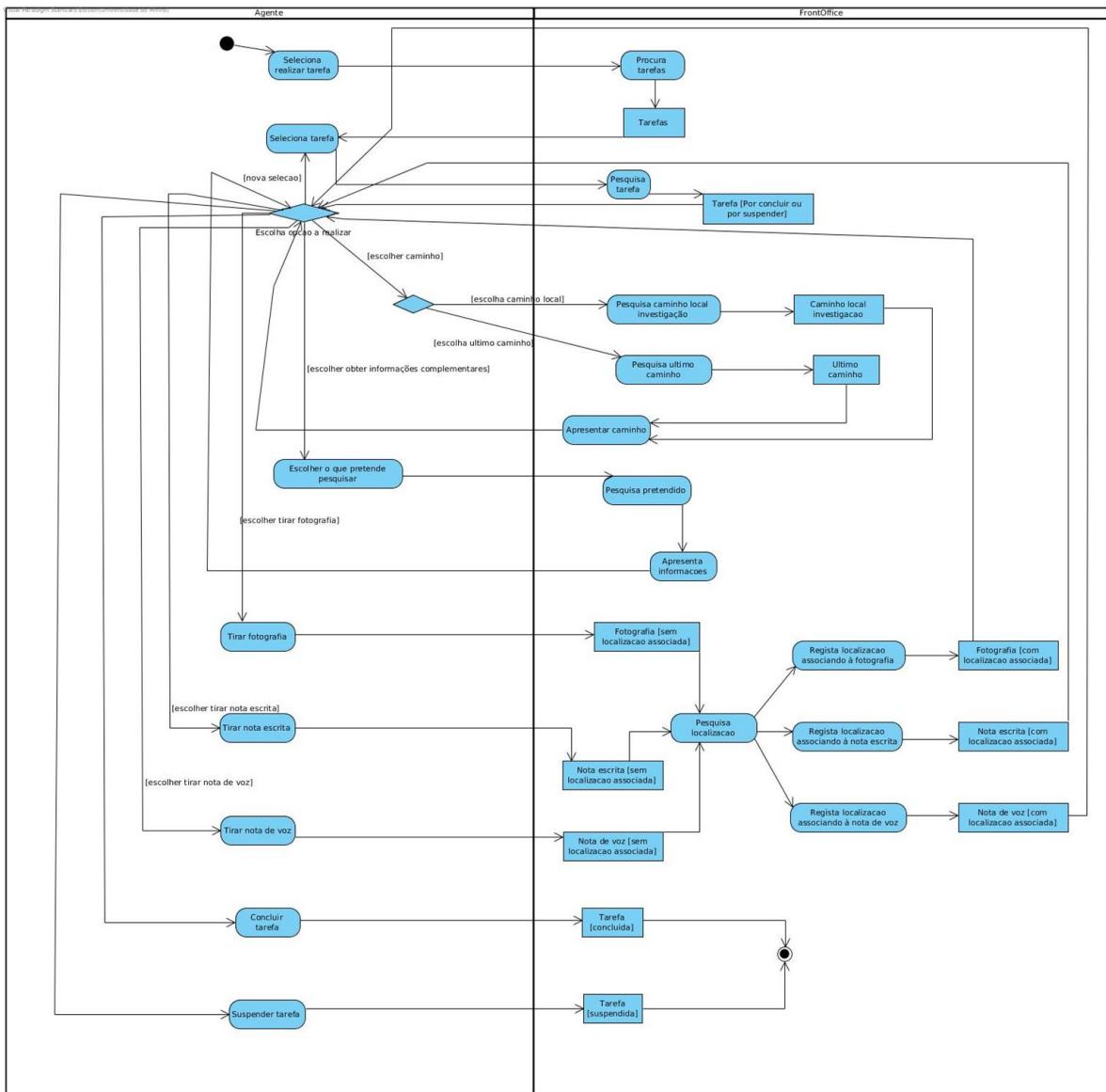


Figura 17 Diagrama de Atividade "Realizar tarefa"

Certas ações podem suscitar algumas dúvidas. Assim, procedeu-se à descrição de algumas ações, nomeadamente as mais importantes. Desta forma, seguem-se alguns exemplos de descrições de ações presente no diagrama de atividades anteriormente exemplificado.

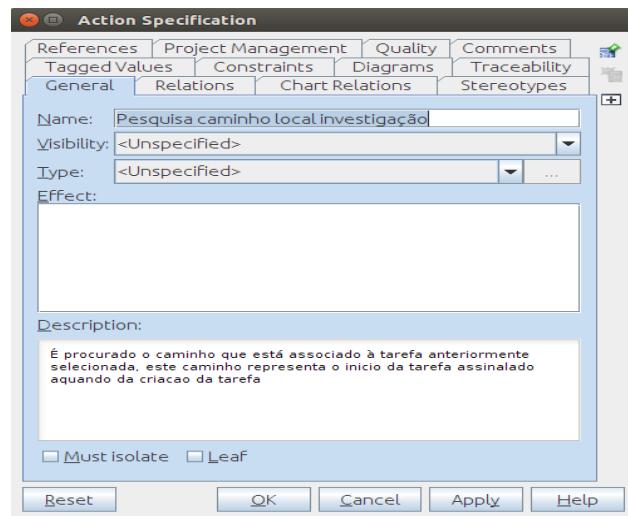


Figura 18 Descrição da ação "Pesquisa caminho local investigação"

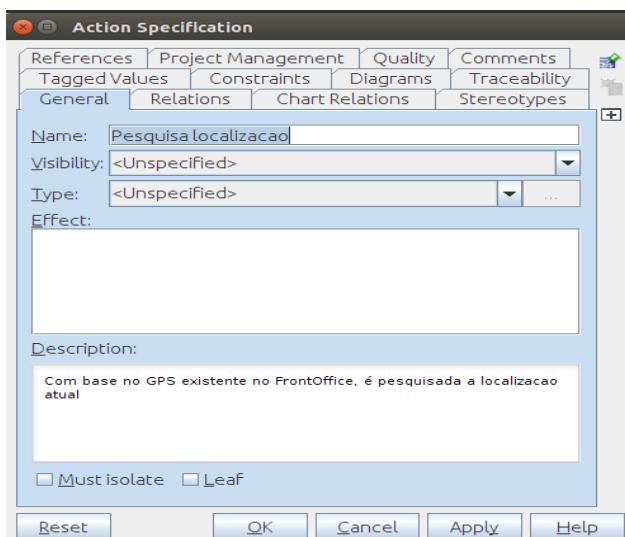


Figura 20 Descrição da ação "Pesquisa localização"

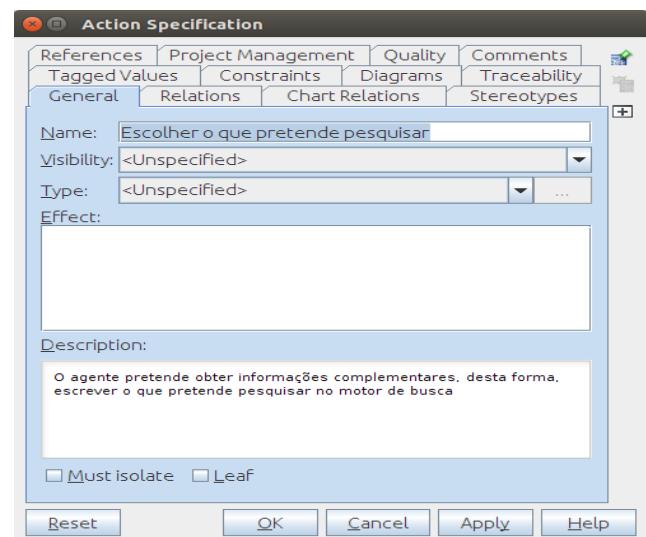


Figura 19 Descrição da ação "Escolher o que pretende pesquisar"

3.1.4. Modelo de Domínio

Para ilustrar os conceitos e as relações entre estes que surgem no contexto do negócio das investigações dos detetives foi construído o Modelo de Domínio abaixo apresentado. Este modelo representa um esquema conceptual do problema em questão e nele podem ser vistas as entidades que têm participação no negócio da empresa, bem como alguns dos seus atributos e papéis que estas desempenham. Como tal, neste modelo não são visíveis funcionalidades concretas que o sistema deve apresentar, pelo que estas são abordadas nas secções próprias.

Na imagem abaixo pode ser visto o Modelo de Domínio mencionado.

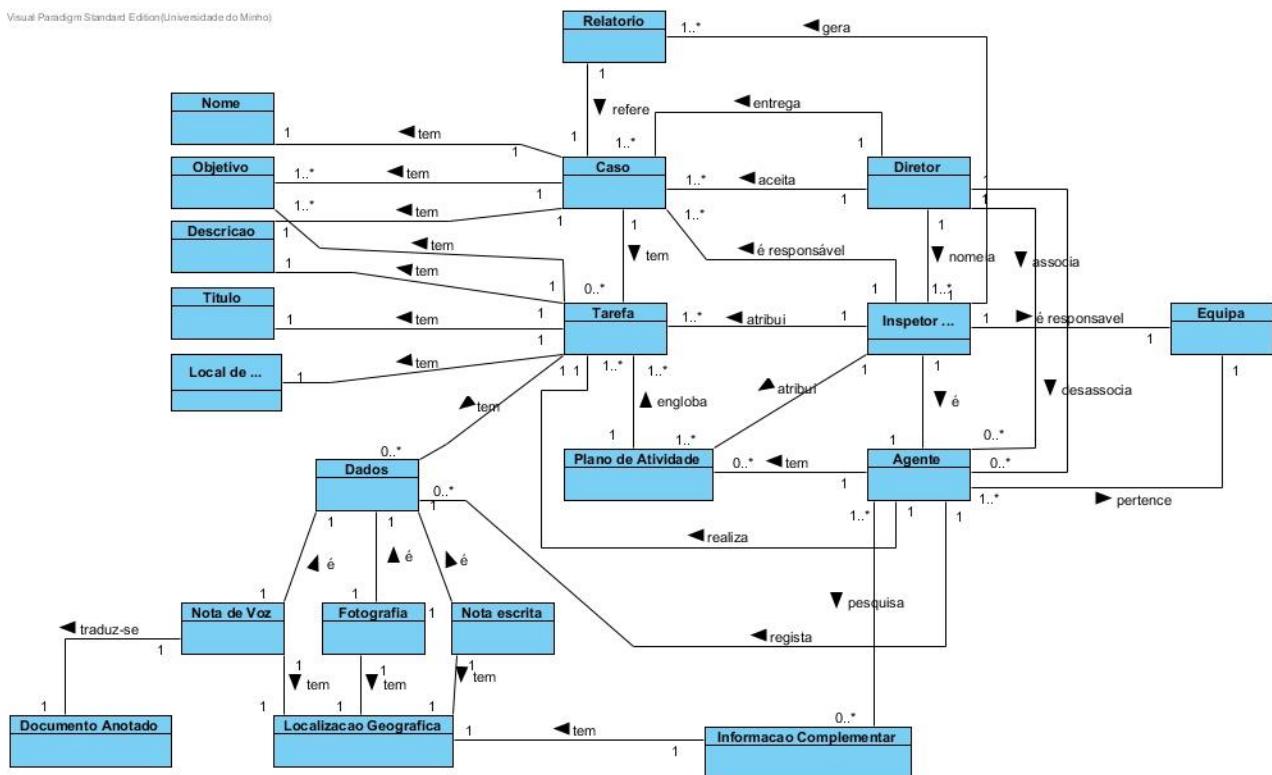


Figura 21 Modelo de Domínio

Tal como se pode ver na imagem acima, as três principais entidades da hierarquia – Diretor, Inspetor-Chefe e Agente – encontram-se representadas no Modelo de Domínio.

O Diretor é o responsável por aceitar os casos que a empresa de detetives irá investigar e por entregar a sua responsabilidade a um Inspetor-Chefe (mencionado no requisito 5). Como tal, estas relações foram representadas através da ligação da entidade Diretor a Caso. Foi utilizado o mesmo raciocínio para as demais entidades que fazem parte do domínio da empresa de detetives criando-se, desta forma, ligações entre estas de maneira a representar as relações ocorridas.

No modelo também podem ser visualizados os atributos que compõem as diferentes entidades (como se pode constatar pelo requisito 3). A título ilustrativo, considere-se a entidade Tarefa. Esta entidade é caracterizada por um conjunto de objetivos, uma descrição, um título, um local de investigação e pelos dados recolhidos durante a execução da tarefa. Como tal, esta associação é representada no Modelo de Domínio através de ligações entre a entidade Tarefa e os atributos respetivos.

O requisito número 4 aponta para o facto de que as tarefas atribuídas a um Agente constituem um plano de atividades. Esta necessidade de englobar as tarefas num plano de atividades é descrita no modelo acima.

Os requisitos 7 e 8 expressam a inclusão de Agentes em equipas lideradas por um determinado Inspetor-Chefe. Esta inclusão dos Agentes numa equipa e liderança desta última por um Inspetor-Chefe são aspetos que também se encontram retratados no Modelo de Domínio.

3.1.5. Diagrama de Classes

De seguida, é apresentado o diagrama de classes. Este diagrama apresenta todas as classes, os seus respetivos atributos e métodos. Além disto, também são especificadas as relações que as classes têm umas com as outras. Para concluir, também é especificado nas relações o tipo de associação (agregação ou composição).

Para a realização deste diagrama foram analisados os requisites e o modelo de domínio.

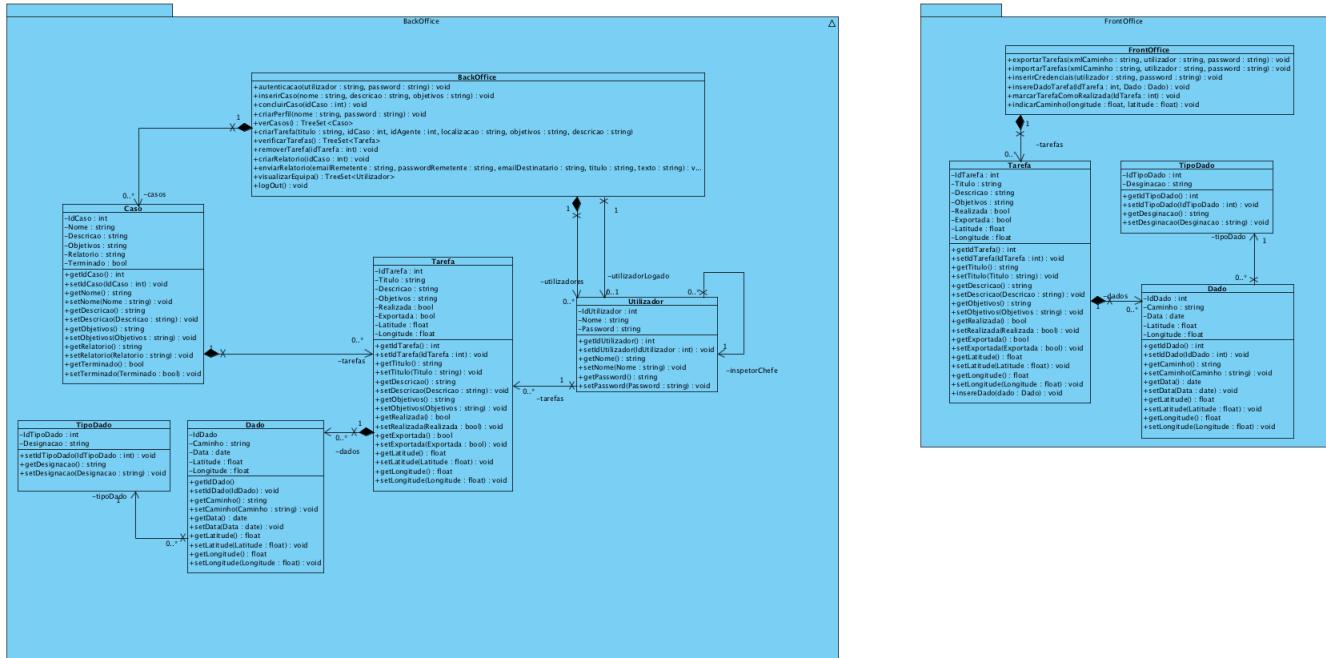


Figura 22 Diagrama de Classes

4. Base de Dados

4.1. Modelo Conceitual

4.1.1. Identificar os tipos de entidades

Após a leitura dos requisitos foram identificadas seis entidades:

- A entidade **Utilizador** representa todas as pessoas que irão interagir com a base de dados: o Diretor, o Inspetor-Chefe e o Agente (mencionado no requisito 1);
- Um **Caso** representa um pedido de investigação (requisito 2). Os casos, tal como mencionado nos requisitos, apenas pode ser inserido por um utilizador com permissões de Diretor (requisito 5.1);
- A entidade **Tarefa** engloba todas as informações das atividades que são realizadas pelos Agente (detalhadas na seção 4.1.3.);

Nome da entidade	Descrição	Sinónimos	Ocorrência
Utilizador	Utilizadores do sistema.	User, Usuário, Membro.	Uma vez por cada utilizador do sistema.
Caso	Casos aceites pela empresa.	Situação, investigação, circunstância.	Sempre que o diretor da empresa aceita um caso, ele é inserido no sistema
Tarefa	Tarefas atribuídas aos Agentes.	Objetivo, atividade, função, missão.	Cada Inspetor-Chefe delega várias tarefas aos seus Agentes.

Tabela 1 Tabela de entidades

4.1.2. Identificar tipos de relacionamento

Após terem sido identificadas as entidades presentes na base de dados, procedeu-se à recolha dos relacionamentos que existem entre estas. A partir da análise de requisitos podemos obter informações sobre a forma como as entidades especificadas se relacionam entre si.

Entre as entidades **Utilizador** e **Caso** existe um relacionamento denominado por “está entregue”. Este relacionamento surge da necessidade de atribuir a responsabilidade de um determinado caso a um Inspetor-Chefe (que é um Utilizador) conforme se pode constatar pelo requisito de utilizador número 5. Uma vez que se poderá entregar a responsabilidade

de vários casos a um Inspetor-Chefe e cada um dos casos apenas terá um responsável associado, a cardinalidade do relacionamento é de 1:N. Não é obrigatório que um Utilizador seja responsável por um caso mas um caso deve ter a sua responsabilidade atribuída a um Utilizador obrigatoriamente.

A entidade **Utilizador** possui um relacionamento recursivo “Pertence ao IC” (IC significa **Inspetor-Chefe**). Como se pode constatar pelo requisito 8, um Inspetor-Chefe é responsável por um conjunto de Agentes e, como tal, surge o relacionamento mencionado. Visto que um Inspetor-Chefe terá a seu cargo uma equipa de Agentes e cada Agente da empresa apenas estará integrado numa equipa (requisito 13) e, portanto, apenas às ordens de um Inspetor-Chefe, a cardinalidade deste relacionamento é de 1:N.

Um **Utilizador** relaciona-se com a entidade **Tarefa** através do relacionamento “Associado”. Este relacionamento surge da necessidade expressa no requisito número 4 em que se enuncia que uma dada tarefa é delegada a um Agente. A cardinalidade deste relacionamento é de 1:N pois um Utilizador (Agente) pode estar associado a várias tarefas, sendo, no entanto, cada Tarefa apenas atribuída a um Utilizador. Um Utilizador não tem forçosamente de ter tarefas associadas mas uma Tarefa deve obrigatoriamente estar associada a um Utilizador.

A entidade **Tarefa** relaciona-se com a entidade **Caso** através do relacionamento “Tem”. Como se pode constatar pelo requisito 3, é necessário associar as tarefas aos casos a que estas dizem respeito, surgindo desta forma o relacionamento mencionado entre estas duas entidades. A cardinalidade do relacionamento é de 1:N uma vez que uma Tarefa tem, obrigatoriamente, de fazer referência ao Caso a que diz respeito e um Caso pode ter várias tarefas associadas.

Entidade	Cardinalidade	Relacionamento	Entidade	Descrição
Caso	N:1	Está entregue	Utilizador	N casos estão entregues a 1 utilizador
Utilizador	N:1	Pertence ao IC	Utilizador	N utilizadores (Agentes) estão às ordens de 1 utilizador (Inspetor-Chefe)
Utilizador	1:N	Associado	Tarefa	1 utilizador está associado a N tarefas
Tarefa	N:1	Tem	Caso	N tarefas têm (estão associadas) um caso.

Tabela 2 Tabela de relacionamentos

4.1.3. Identificar e associar atributos com os tipos de entidades e relacionamentos

Neste tópico serão abordados todos os atributos que compõem as entidades e os relacionamentos. A escolha dos atributos foi realizada com base numa análise dos requisitos a fim de detetar todos os atributos necessários para representar cada entidade e relacionamento no sistema.

4.1.3.1. Associação entre atributos e entidades/relacionamentos

Entidade	Atributo	Descrição	Tipo e tamanho	Nulo	V.P.D
Utilizador	<u>Código</u>	Identifica inequivocamente um utilizador	Valor inteiro positivo	Não	-
	Nome	Nome do utilizador	5-30 caracteres variáveis	Não	-
	<i>Password</i>	Palavra-chave do utilizador	5-30 caracteres variáveis	Não	-
Tarefa	<u>Código</u>	Identifica inequivocamente uma tarefa	Valor inteiro positivo	Não	-
	Título	Título da tarefa	5-30 caracteres variáveis	Não	-
	Descrição	Descrição da tarefa	Texto	Não	-
	Objetivos	Objetivos a alcançar com a realização da tarefa	Texto	Não	-
	Realizada	Indica se a tarefa já foi realizada	Boolean	Não	False
	Exportada	Indica se a tarefa já foi exportada	Boolean	Não	False
	Localização	Localização de uma tarefa	-	Não	-

	Latitude	Latitude da localização da tarefa	Valor real	Não	-
	Longitude	Longitude da localização da tarefa	Valor real	Não	-
	Dado	Dado de uma tarefa	5-30 caracteres variáveis	Não	-
	Tipo	Tipo de dados (fotografias, registo de voz, etc.)	5-30 caracteres variáveis	Não	-
	Data	Data em que o dado foi recolhido	Data	Não	-
	Localização	Localização de um dado	-	Não	-
	Latitude	Latitude da localização do dado	Valor real	Não	-
	Longitude	Longitude da localização do dado	Valor real	Não	-
Caso	<u>Código</u>	Identifica inequivocamente um caso	Valor inteiro positivo	Não	-
	Nome	Nome do caso	5-30 caracteres variáveis	Não	-
	Descrição	Descrição do caso	Texto	Não	-
	Objetivos	Objetivos a alcançar com a realização do caso	Texto	Não	-
	Relatório	Relatório final do caso	3-40 caracteres variáveis	Sim	NULL
	Terminado	Indica se o caso está concluído	Boolean	Não	False

Tabela 3 Tabela de atributos de entidades e relacionamentos

4.1.4. Determinar domínios dos atributos

De seguida é apresentado o domínio de cada atributo presente no modelo conceptual. O domínio consiste no conjunto de valores que cada atributo pode tomar.

Entidade: Utilizador

- Código: Atributo que serve para identificar um utilizador. Corresponde a um inteiro positivo;
- Nome: Corresponde ao nome do utilizador (dizer que inclui tb o apelido conforme os requisitos, ou tem de haver um a parte???). É uma *string* até 30 caracteres;
- Password: *String* que corresponde á palavra-chave do utilizador. Pode conter até 30 caracteres;

Entidade: Tarefa

- Código: Atributo que serve para identificar uma tarefa. Corresponde a um inteiro positivo;
- Título: Atributo que corresponde ao título da tarefa. Corresponde a uma *string* com até 30 caracteres;
- Descrição: Corresponde a um texto com a descrição da tarefa;
- Objetivos: O atributo “objetivos” corresponde a um texto com os objetivos a alcançar com a realização da tarefa;
- Realizada: O atributo “realizada” corresponde a um *boolean* que indica se a tarefa já foi realizada ou não;
- Exportada: O atributo “exportada” corresponde a um *boolean* que indica se a tarefa já foi exportada ou não;
- Dado: O “dado” corresponde a uma informação recolhida sobre uma tarefa;
- TipoDado: Este atributo corresponde á designação do tipo de dado (fotografia, por exemplo). É uma *string* até 30 caracteres;
- Localização: Corresponde à localização geográfica. A latitude e a longitude são representadas em graus;
- Latitude: A “latitude” é um *double* que pode conter valores entre -90 a 90;
- Longitude: A “longitude” é um *double* que pode conter valores entre -180 a 180;

Entidade: Caso

- Código: Atributo que serve para identificar um caso. Corresponde a um inteiro positivo.
- Nome: O atributo “nome” corresponde ao título do caso. É uma *string* com até 30 caracteres;
- Descrição: Este atributo corresponde a um texto com uma descrição do caso;

- Objetivos: Os “objetivos” são um texto com todos os objetivos/ metas a alcançar com a realização do caso;
- Relatório: Este atributo é uma *string* que corresponde ao caminho em que o relatório está armazenado;
- Terminado: Este atributo consiste num *boolean* que indica se o caso está concluído ou não.

4.1.5. Determinar chaves primárias, candidatas e alternativas

De maneira a identificar unicamente a ocorrência das entidades foi necessário determinar os atributos chave.

Nenhuma das entidades continha um atributo que identificasse inequivocamente um registo. Como tal, foi necessária atribuir a cada uma destas entidades um atributo **Código**. Este atributo consiste num identificador numérico que identifica um registo de forma única, satisfazendo assim a noção de chave primária.

Com base no que foi explicado no parágrafo anterior não existem assim chaves candidatas nem alternativas.

4.1.6. Desenho do diagrama ER

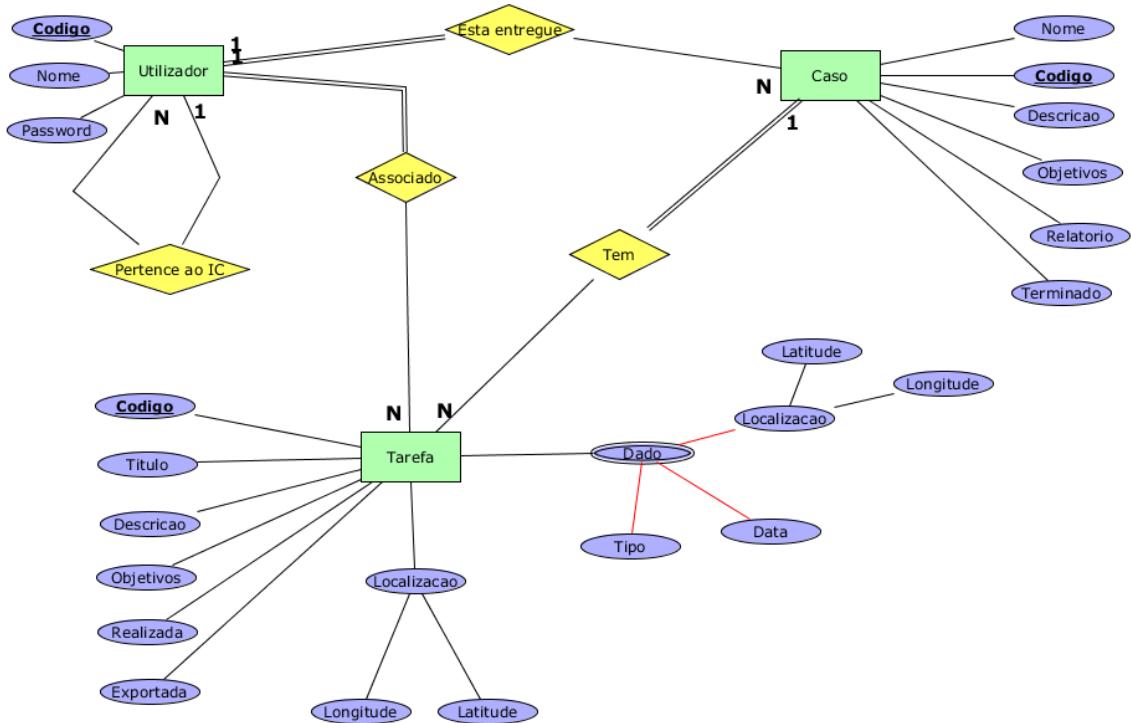


Figura 23 Modelo Conceptual

4.1.7. Revisão do modelo de dados com o utilizador

Após a conceção do modelo conceptual, este foi revisto junto do utilizador. Dado que nenhum erro ou problema foi encontrada, este modelo foi aceite.

4.2. Modelo Lógico

4.2.1. Passagem do modelo conceptual para o modelo lógico

4.2.1.1. Entidades

Cada uma das entidades presentes no modelo conceitual (Utilizador, Tarefa e Caso) são entidades fortes e como tal traduzem-se na criação de três tabelas.

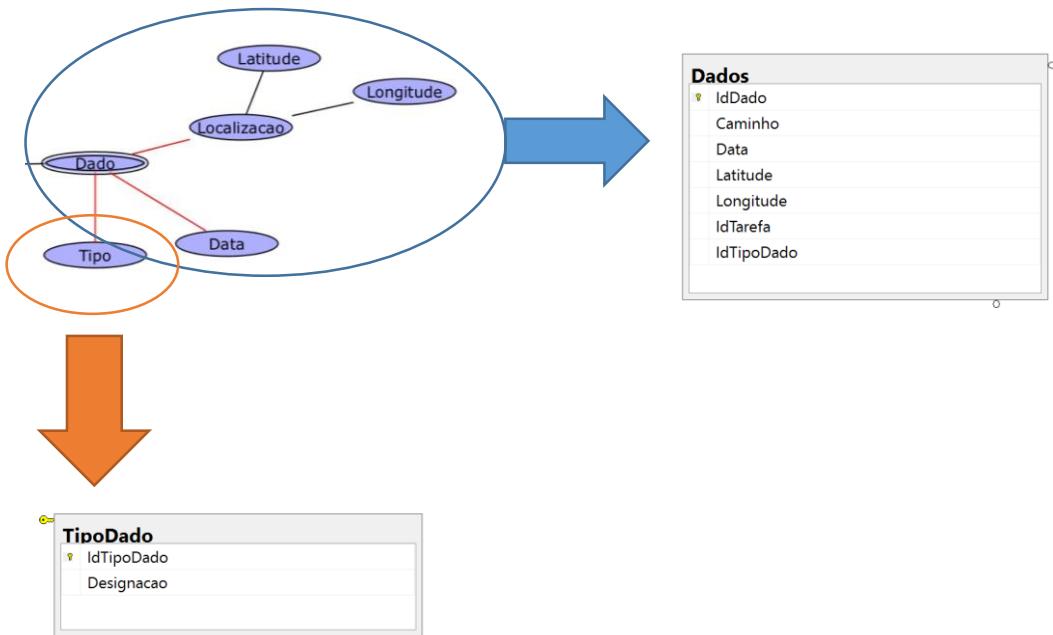
4.2.1.2. Relacionamentos

No modelo conceitual estavam representados quatro relacionamentos, todos eles 1:N. Estes relacionamentos foram mantidos no modelo lógico através da inclusão das respetivas chaves estrangeiras na tabela de maior cardinalidade.

4.2.1.3. Atributos

O atributo multi-valor “Dado” foi traduzido numa tabela. Para evitar a redundância de dados e garantir a consistência da informação na base dados, foi criada uma outra tabela denominada “TipoDado” que conterá os diferentes tipos de dados existentes, por exemplo: fotografia, notas escritas, etc

Em baixo é apresentada uma imagem que ilustra o raciocínio explicado no último parágrafo.



Com o surgimento destas duas tabelas, surgiram mais dois relacionamentos. Com a nova tabela “Dado” surgem os relacionamentos de 1:N com as tabelas “Tarefa” e “TipoDado” como apresentado na imagem em baixo:



Figura 24 Surgimento de novas tabelas

Como se pode constatar na imagem acima, a chave estrangeira da tabela “TipoDado” está presente na tabela “Dados” bem como a chave estrangeira da tabela “Tarefa” “idTarefa”.

A tabela TipoUtilizador que não estava inicialmente prevista foi incluída no modelo lógico de maneira a relacionar cada utilizador (presente na tabela Utilizador) com a sua função (Diretor, Inspetor-Chefe ou Detetive). Como tal, cada registo da tabela Utilizador possui uma chave estrangeira (IdTipoUtilizador) que referencia um registo da tabela TipoUtilizador que indica a função (Descricao) ocupada pela pessoa do registo em causa.

A inclusão desta tabela é demonstrada na imagem final do modelo lógico.

Todos os restantes atributos das entidades do modelo conceitual foram normalmente traduzidos para atributos das respetivas tabelas no modelo lógico.

4.2.2. Validação do modelo lógico através de normalização

Após analisarmos as dependências funcionais de cada relação verificamos que as tabelas respeitam as três primeiras regras de normalização e, portanto, encontram-se normalizadas até à Terceira Forma Normal da Normalização.

4.2.3. Elaboração e validação do esquema lógico da base de dados

Após a realização do modelo lógico e da validação feita na secção anterior, conclui-se que este se encontra corretamente elaborado e validado.

Na imagem seguinte pode ser visualizado o esquema lógico construído.

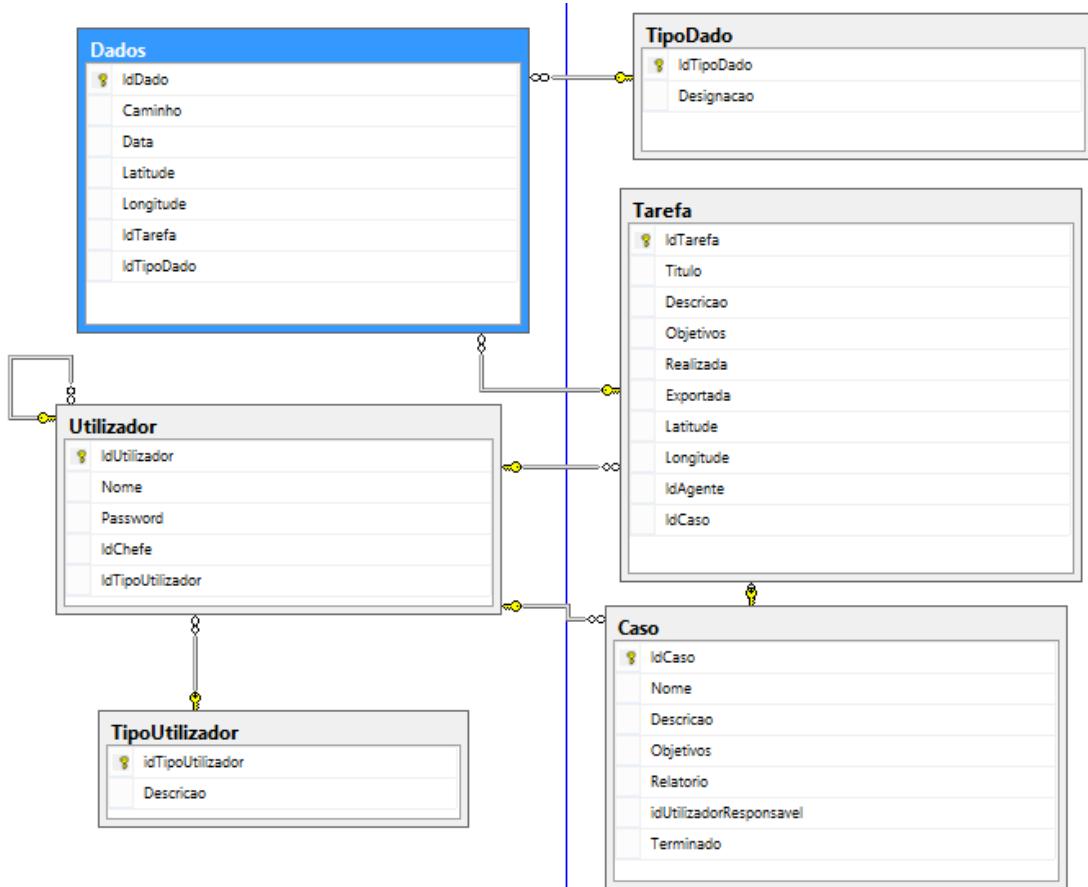


Figura 25 Modelo Lógico

4.2.4. Definição do tamanho inicial da base de dados e análise do seu crescimento futuro

Esta base de dados foi projetada para satisfazer todos os requisitos, contudo uma eventual extensão do problema não compromete o modelo lógico. Por exemplo, se o cliente necessitasse de inserir as viaturas a cargo da empresa e os dados originados a partir da utilização por parte dos Agentes (data de inicio e fim de aluguer, distâncias percorridas, quem utilizou a viatura, casos a que estiveram associados, etc.) esta implementação levaria a criação de novas tabelas quem seriam facilmente integradas no modelo aqui apresentado, não comprometendo a integridade do modelo até então criado.

4.2.5. Revisão do modelo lógico final com os futuros utilizadores

De maneira a verificar se todos os requisitos se encontram cumpridos, procedeu-se a uma revisão do modelo com o cliente com o intuito de assegurar que este o considera uma representação fidedigna do que é pretendido. Outro objetivo desta revisão foi permitir ao cliente detetar eventuais problemas ou falhas que possam causar complicações no futuro.

Após esta revisão, concluiu-se que o problema em questão se encontrava representado corretamente no modelo lógico construído.

4.3. Modelo físico

4.3.1. Acesso à base de dados

4.3.1.1. Logins e users

Nesta fase do trabalho foram criados vários *logins* de forma a definir autenticações no SQL Server e também foram criados *users* de forma a serem definidas as permissões de utilização sobre a base de dados.

Os três *logins* criados são: Diretor, IC e Agente, sendo que cada *login* tem uma *password* de acesso à BD distinta. São criados estes *logins* uma vez existirem apenas estas três entidades no nosso sistema que podem ter acesso à base de dados. A figura que se segue representa a criação dos *logins*:

```

create login Diretor with password = 'diretor',
default_database = [LI4-Agentes];

create login IC with password = 'inspetorchefe',
default_database = [LI4-Agentes];

create login Agente with password = 'agente',
default_database = [LI4-Agentes];

```

Figura 26 - Criação de *logins*

Em relação a utilizadores da base de dados foram também criados três *users* diferentes de forma a representar as três entidades do nosso projeto: Diretor, IC e Agente. A cada *user* é associado um determinado *role* criado com as permissões que o utilizador tem sobre a BD. Acerca dos *roles* criados é falado no ponto 4.3.1.2. A figura que se segue representa a criação dos três *users* criados:

```

create user Diretor for login Diretor;

create user IC for login IC;

create user Agente for login Agente;

```

Figura 27 - Criação de *users*

4.3.1.2. Roles

Uma vez criados os *users* é necessário a criação de *roles* com as diversas permissões sobre a base de dados e associar os *users* a um *role*.

Foi criado três roles distintos: LI4_Diretor, LI4_IC e LI4_Agente. O *role* LI4_Diretor, tal como o nome sugere, contém as permissões sobre a base de dados para um utilizador que seja um Diretor. Este tipo de *user* existe apenas no contexto do *BackOffice* sendo que as ações que pode realizar são: criar casos, criar perfis e visualizar equipas. Todas estas ações estão mencionadas na secção 2 - Requisitos. Desta forma, as permissões (*Grants*) de acesso que os utilizadores Diretor têm, relativamente às tabelas e *views*, são:

- **Select:** Casos, Utilizador, viewEquipas, viewInspetoresChefe;
- **Insert:** Casos, Utilizador;
- **Update:** Casos, Utilizador;
- **Delete:** Casos;

O *role* LI4_IC contém as permissões sobre a base de dados para um *user* que seja Inspetor-Chefe. Um inspetor-Chefe no contexto do *BackOffice* tem bastantes ações que pode

realizar tal como foi identificado e mencionado ao longo do projeto. Desta forma, as permissões (*Grants*) de acesso que os *users* IC têm, relativamente às tabelas e views são:

- **Select:** Casos, Dados, Tarefa, Utilizador, viewEquipas;
- **Insert:** Tarefa;
- **Update:** Casos, Dados, Tarefa;
- **Delete:** Tarefa.

Relativamente ao último *role* LI4_Agentes, este contém as permissões sobre a base de dados para um *user* Agente.

Desta forma, as permissões (*Grants*) de acesso que os utilizadores Agente têm, relativamente às tabelas e views, são:

- **Select:** Tarefas, Casos, Utilizador
- **Insert:** Dados
- **Update:** Tarefas

Desta maneira, cumpre-se o requisito número 14 que diz que um Agente deve poder ter acesso aos casos em que está inserido (tabela Casos) e deve poder consultar o seu plano de atividades (na tabela Tarefas é onde estão indicadas as tarefas que este tem de realizar).

De maneira a que o Agente possa também marcar as suas tarefas como exportadas e realizadas, tem também permissão de Update sobre a tabela Tarefas.

De acordo com o requisito 19.1, o Agente deve poder exportar para o *BackOffice* os dados que recolheu durante o seu trabalho de campo e, como tal, é-lhe dada permissão de inserção nesta tarefa.

4.3.2. *Stored Procedures*

Durante a conceção da aplicação foi surgindo necessidade de com base num dado argumento obter informação da base de dados de forma específica. Deste modo, foram criados diversos *Stored Procedures* que permitem esta obtenção de informação. Os *Stored Procedures* são associados a um determinado *role* existente permitindo assim, a sua utilização, apenas a *users* que sejam membros desse *role*.

O *role* LI4_Diretor não tem nenhum *Stored Procedure* associado, no entanto, o *role* LI4_IC tem um elevado número de *Stored Procedures* associados. Foi criado um *Stored Procedure* denominado de AgentesEnvolvidos que consiste em dado um idCaso devolve os registos com a informação do id e o nome dos utilizadores que tenham associadas tarefas a esse caso. Este *Stored Procedure* foi utilizado aquando da criação de relatórios, uma vez que permitiu ter todos os agentes que realizaram tarefas de um dado caso, sendo assim inserida no corpo dos relatórios. Na seguinte figura é apresentada a implementação do *Stored Procedure* AgentesEnvolvidos:

```

CREATE PROCEDURE [dbo].[AgentesEnvolvidos] (@idCaso int)
AS
BEGIN
    SELECT dbo.Utilizador.IdUtilizador, dbo.Utilizador.Nome
    FROM dbo.Utilizador INNER JOIN
        dbo.Tarefa ON dbo.Utilizador.IdUtilizador = dbo.Tarefa.IdAgente
    WHERE dbo.Tarefa.IdCaso = @idCaso
END

```

Figura 28 - Criação do Stored Procedure AgentesEnvolvidos

Outro *Stored Procedure* implementado foi o CasosAtivos que consiste em dado um id de um Inspetor-Chefe devolver regtos com a seguinte informação: IdCaso, Nome, Descrição, Objetivos, Relatorio e Terminado da tabela Casos no qual o responsável pelo caso seja igual ao passado como argumento. Basicamente, este *Stored Procedure* permite obter informação sobre os casos no qual um dado Inspetor-Chefe é responsável. Este *Stored Procedure* foi utilizado em diversos momentos da conceção da aplicação, tais como aquando da conclusão de um caso, da criação de relatórios, da criação de tarefas e da visualização de casos. A título de exemplo, foi utilizado o CasosAtivos na conclusão de um caso uma vez que foi necessário a obtenção dos casos por concluir do Inspetor-Chefe logado. Na seguinte figura é apresentada a implementação do *Stored Procedure* CasosAtivos:

```

|CREATE PROCEDURE [dbo].[CasosAtivos] (@idIC int)
|AS
|BEGIN
|    SELECT IdCaso, Nome, Descricao, Objetivos, Relatorio, Terminado
|    FROM dbo.Caso AS C
|    WHERE (idUtilizadorResponsavel = @idIC)
|END

```

Figura 29 - Criação do Stored Procedure CasosAtivos

O *Stored Procedure* dadosRelatorioAgente consiste em que dado um id de uma Agente e um id de um Caso, devolve os regtos com a informação do título de uma tarefa e o número de dados relacionados com essa tarefa no qual o IdCaso da tarefa seja igual ao passado como argumento e o IdAgente da tarefa seja igual ao passado como argumento do *Stored Procedure*. Este *Stored Procedure* foi utilizado na criação de relatórios na inserção da informação obtida no corpo do relatório. Na seguinte figura é apresentada a implementação do *Stored Procedure* dadosRelatorioAgente:

```

|CREATE PROCEDURE [dbo].[dadosRelatorioAgente] (@idAgente int, @idCaso int)
AS
|BEGIN
|   SELECT dbo.Tarefa.Titulo, COUNT(dbo.Dados.IdDados) AS NumeroDados
|   FROM dbo.Tarefa INNER JOIN
|       dbo.Dados ON dbo.Tarefa.IdTarefa = dbo.Dados.IdTarefa
|   WHERE (dbo.Tarefa.IdCaso = @idCaso) AND (dbo.Tarefa.IdAgente = @idAgente)
|   GROUP BY dbo.Tarefa.Titulo
|
|END

```

Figura 30 - Criação do Stored Procedure dadosRelatorioAgentes

Outro *Stored Procedure* criado foi o tarefasDosAgentesDeUmIC que consiste em que dado um id de um Inspetor-Chefe devolve a seguinte informação: IdUtilizador, Nome do utilizador, IdTarefa, Caminho do dado e tipo do dado. Este *Stored Procedure* foi utilizado aquando da transformação de ficheiros áudio em ficheiros anotados, permitindo mostrar esta informação antes de se proceder à transformação. Na seguinte figura é apresentada a implementação do *Stored Procedure* tarefasDosAgentesDeUmIC:

```

|CREATE PROCEDURE [dbo].[tarefasDosAgentesDeUmIC](@idIC int)
AS
|BEGIN
|   SELECT dbo.Utilizador.IdUtilizador, dbo.Utilizador.Nome, dbo.Tarefa.IdTarefa, dbo.Dados.Caminho, dbo.Dados.IdDados, dbo.Dados.IdTipoDado
|   FROM dbo.Utilizador INNER JOIN
|       dbo.Utilizador AS Utilizador_1 ON dbo.Utilizador.IdChefe = Utilizador_1.IdUtilizador INNER JOIN
|       dbo.Tarefa ON dbo.Utilizador.IdUtilizador = dbo.Tarefa.IdAgente INNER JOIN
|       dbo.Dados ON dbo.Tarefa.IdTarefa = dbo.Dados.IdTarefa
|   WHERE (Utilizador_1.IdUtilizador = @idIC)
|END

```

Figura 31 – Criação do Stored Procedure tarefasDosAgentesDeUmIC

Um outro *Stored Procedure* implementado denomina-se por TarefasIC e consiste em dado um id de um Inspetor-Chefe devolver registos com a seguinte informação: id, titulo, descrição, objetivos, latitude, longitude da tarefa, nome do caso, id e nome do agente responsável pela tarefa e informação do estado de realização e exportação da tarefa em que os Inspetores-Chefes dos utilizadores sejam o passado como argumento. É utilizado este *Stored Procedure* quando se pretende visualizar as tarefas dos agentes associados a um determinado Inspetor-Chefe, podendo desta forma visualizar o estado das tarefas. Na seguinte figura é apresentada a implementação do *Stored Procedure* TarefasIC:

```

|CREATE PROCEDURE [dbo].[TarefasIC] (@idIC int)
AS
|BEGIN
|   SELECT dbo.Tarefa.IdTarefa, dbo.Tarefa.Titulo, dbo.Tarefa.Descricao,
|       dbo.Tarefa.Objetivos, dbo.Tarefa.Latitude, dbo.Tarefa.Longitude, dbo.Caso.Nome AS NomeCaso, dbo.utilizador.IdUtilizador AS IdAgente,
|       dbo.Utilizador.Nome AS Agente, dbo.Tarefa.Realizada, dbo.Tarefa.Exportada
|   FROM dbo.Tarefa INNER JOIN
|       dbo.Utilizador ON dbo.Tarefa.IdAgente = dbo.Utilizador.IdUtilizador INNER JOIN
|       dbo.Caso ON dbo.Tarefa.IdCaso = dbo.Caso.IdCaso
|   WHERE (dbo.Utilizador.IdChefe = @idIC)
|END

```

Figura 32 - Criação do Stored Procedure TarefasIC

Por fim, foi implementado um último *Stored Procedure* ao qual foi associado ao *role* LI4-IC. Este último *Stored Procedure* denomina-se de TarefasNaoSincronizadas e consiste em dado um id de um Inspetor-Chefe devolve registos com as seguintes informação: id, título, descrição, objetivos, latitude e longitude da tarefa, nome do caso e nome do Agente responsável pela tarefa em que os Inspetores-Chefes dos utilizadores sejam o mesmo do passado como argumento. O *Stored Procedure* TarefasNaoSincronizadas é usado quando um IC pretende remover uma determinada tarefa, visto apenas ser possível a remoção de tarefas associadas a agentes da equipa do IC. Sendo que através deste *Stored Procedure* é possível obter as informações dessa tarefas. Na seguinte figura é apresentada a implementação do *Stored Procedure* TarefasNaoSincronizadas:

```

CREATE PROCEDURE [dbo].[TarefasNaoSincronizadas] (@idInspetorChefe int)
AS
BEGIN
    SELECT dbo.Tarefa.IdTarefa, dbo.Tarefa.Titulo, dbo.Tarefa.Descricao, dbo.Tarefa.Objetivos, dbo.Tarefa.Latitude,
    dbo.Tarefa.Longitude, dbo.Caso.Nome AS NomeCaso, dbo.Utilizador.Nome
    FROM dbo.Tarefa INNER JOIN
        dbo.Utilizador ON dbo.Tarefa.IdAgente = dbo.Utilizador.IdUtilizador INNER JOIN
        dbo.Caso ON dbo.Tarefa.IdCaso = dbo.Caso.IdCaso
    WHERE (dbo.Utilizador.IdChefe = @idInspetorChefe)
END

```

Figura 33 - Criação do Stored Procedure TarefasNaoSincronizadas

4.3.3. Views

As *views* permitem ter acesso a dados de diferentes tabelas na forma pretendida, desta forma, para auxiliar a conceção da aplicação foram criadas vistas e associadas a *roles*.

Tanto um utilizador do tipo Diretor como Inspetor-Chefe têm a possibilidade de ver as equipas, no caso do Diretor, este pode ver todas as equipas. Ao invés, os Inspetores-Chefes apenas podem ver os dados das suas equipas. Estes aspectos já foram anteriormente analisados e mencionados em capítulos anteriores. Esta informação necessária para mostrar aos utilizadores não se encontra apenas numa tabela, deste modo, foi criada uma *view* denominada de viewEquipas que vai agrupar informação de diferentes tabelas e mostrar esta informação por ordem de Inspetor-Chefe. A *view*, depois de criada, foi associada ao *role* LI4-Diretor e LI4-IC. Na seguinte figura é apresentada a implementação da *view*, no qual é possível verificar a informação que cada registo da *view* contém:

```

CREATE VIEW [dbo].[viewEquipas]
AS
SELECT TOP (100) PERCENT dbo.Utilizador.IdUtilizador, dbo.Utilizador.Nome, COUNT(dbo.Tarefa.IdTarefa) AS TarefasDecorrer,
Utilizador_1.Nome AS InspetorChefe, Utilizador_1.IdUtilizador AS IdInspetorChefe
FROM dbo.Utilizador LEFT OUTER JOIN
dbo.Utilizador AS Utilizador_1 ON dbo.Utilizador.IdChefe = Utilizador_1.IdUtilizador LEFT OUTER JOIN
dbo.Tarefa ON dbo.Utilizador.IdUtilizador = dbo.Tarefa.IdAgente
WHERE (dbo.Utilizador.IdTipoUtilizador <> 1)
GROUP BY dbo.Utilizador.IdUtilizador, dbo.Utilizador.Nome, Utilizador_1.Nome, Utilizador_1.IdUtilizador
ORDER BY IdInspetorChefe

```

Figura 34 - Criação da view viewEquipas

Um Diretor tem a possibilidade de criar perfis, sendo que neste processo de criação uma das ações a realizar é a escolha de um Inspetor-Chefe para o perfil criado. Assim, surge a necessidade de criar uma vista que contem os registo com a informação do Id e nome de todos os Inspetores-Chefe presentes na base de dados. Após a criação desta vista, ela foi associada ao *role* LI4-Diretor. A seguinte figura ilustra a implementação desta vista mencionada, ao qual foi denominada de viewInspetoresChefe:

```

CREATE VIEW [dbo].[viewInspetoresChefe]
AS
SELECT Nome, IdUtilizador
FROM dbo.Utilizador
WHERE (IdTipoUtilizador = 3)

```

Figura 35 - Criação da view viewInspetoresChefe

5. Mockups

De seguida são apresentados os *mockups* realizados para cada sistema.

Com esta ilustração pretendemos planejar o aspetto da *interface* com o utilizador, ou seja, da camada de apresentação.

Primeiramente, são apresentados os *mockups* do sistema do *backOffice*. Estes *mockups* apresentam um menu à esquerda em que são apresentadas todas as opções de funcionalidade. Quando uma opção é selecionada, o painel à direita é alterado para a funcionalidade em questão.

Na imagem abaixo pode ser visualizado o *mockup* correspondente à janela de criação de uma tarefa. Nesta funcionalidade, é necessário que se introduzam os dados que compõem uma tarefa (mencionados no requisito 3) e, como tal, o painel à direita disponibiliza os campos necessários ao seu preenchimento.

O mockup mostra a interface de usuário do BackOffice. No lado esquerdo, há uma barra lateral com opções: Inserir caso, Concluir caso, Criar perfil, Ver casos, Criar tarefa (destacada com um fundo verde), Verificar tarefas, Remover tarefa, Criar relatório, Enviar relatório, Visualizar equipa e Logout. No lado direito, para a criação de uma tarefa, há campos para Título (campo com placeholder "Indique o título"), Caso (menu suspenso com "Investigação Seguradora"), Agente (menu suspenso com "João Marco"), Localização (campo com placeholder "Indique a localização") e Objetivos (campo grande para inserir texto). Abaixo desses campos, há um campo para Descrição (campo grande para inserir texto) e um botão "Confirmar" no canto inferior direito.

Figura 36 Mockup Criar Tarefa

Na imagem abaixo pode ser visto o *mockup* correspondente à visualização dos casos.

Como tal, o utilizador deve selecionar o caso que pretende visualizar na *comboBox* destinada para o efeito. Quando um *item* está selecionado, as informações referentes ao caso em questão (mencionadas no requisito 2) são apresentadas na janela. Visto o relatório do caso consistir num documento em formato *PDF*, é disponibilizado um botão “Abrir relatório” que abre o ficheiro pretendido caso este exista (ou seja, caso tenha sido gerado).

De maneira a melhor filtrar os casos que são apresentados para escolha, existe uma *checkbox* que quando está marcada, apenas disponibiliza como opções de escolha os casos que se encontram marcados como concluídos. Quando esta *checkbox* não se encontra selecionada, apenas os casos em curso (que não estão marcados como terminados) são apresentados.

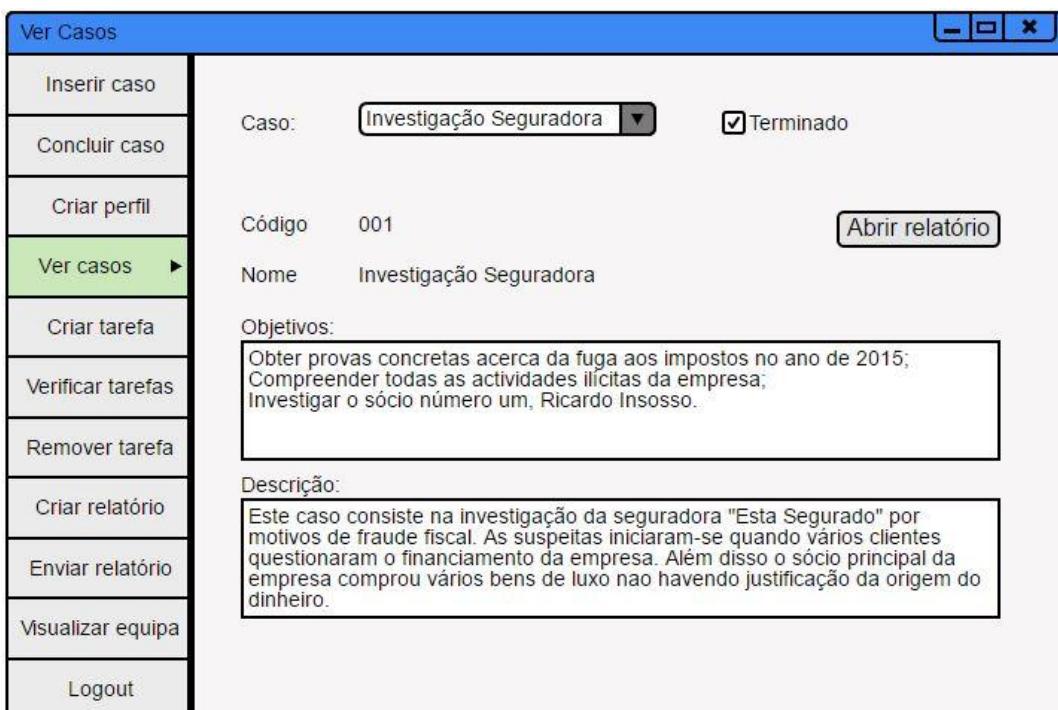


Figura 37 Mockup Ver Casos

O *mockup* seguinte refere-se à visualização de uma equipa de Agentes. Quando a opção correspondente se encontra selecionada no menu à esquerda, é apresentada no painel da direita uma lista dos Agentes que pertencem à equipa do Inspetor-Chefe em questão (ou seja, do Inspetor-Chefe que tem login efetuado no *backOffice*). Se o Diretor selecionar a opção de *Visualizar Equipa*, são apresentados na lista todos os Agentes.

Ver Casos			
	Inserir caso		
	Concluir caso		
	Criar perfil		
	Ver casos		
	Criar tarefa		
	Verificar tarefas		
	Remover tarefa		
	Criar relatório		
	Enviar relatório		
	Visualizar equipa		
	Logout		

▼ Código Agente	▼ Nome Agente	▼ Nr Tarefas a decorrer
001	Alexandre Silva	3
002	Francisco Ribeiro	4
003	Rafael Mota	7
004	Rui Rua	2

Figura 38 Mockup Visualizar Equipa

Em seguida são apresentados os *mockups* referentes ao *frontOffice*.

Quando um Agente inicia o *frontOffice*, a primeira interface que lhe é apresentada é o menu. No menu, tal como é apresentado na imagem a seguir, o Agente pode escolher entre quatro opções:

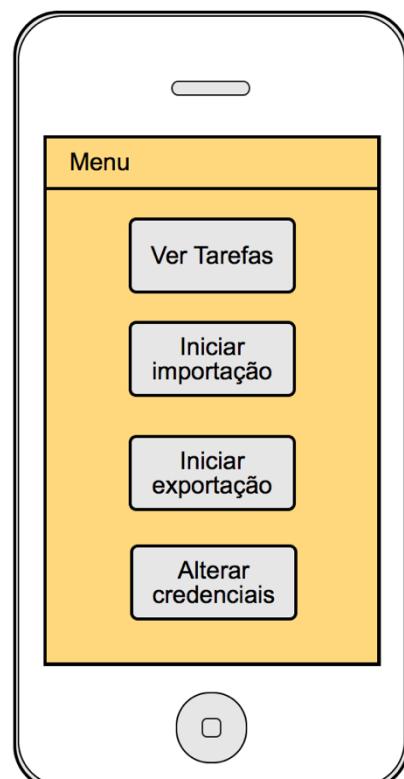


Figura 39 Mockup Menu
frontOffice

Depois do menu, caso o Agente tenha selecionado a opção “Ver Tarefas” é-lhe apresentada a interface que se segue na imagem abaixo. É a partir desta lista de tarefas que um Agente procede para a sua realização:

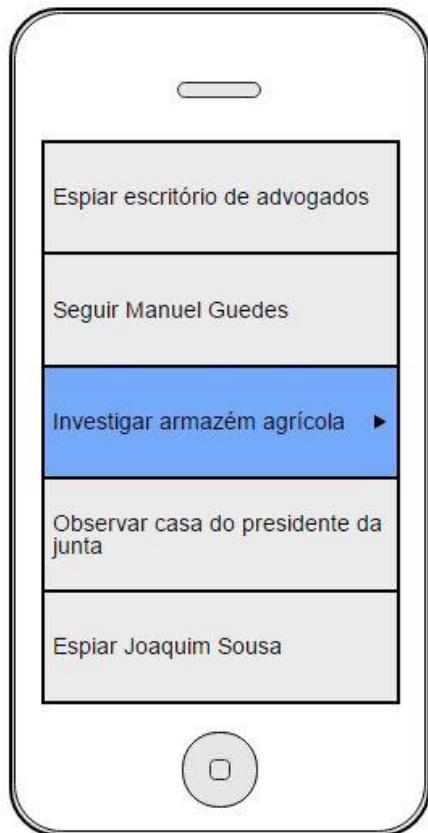


Figura 40 Mockup Ver Tarefas

No *mockup* seguinte é apresentada a *interface* resultante da seleção prévia de uma tarefa por parte do Agente. Após esta ter sido selecionada, o ecrã mostrado abaixo é apresentado. Este contém um menu à esquerda com todas as opções disponíveis para a realização de uma tarefa (conforme indicado nos requisitos 16, 17 e 18). Na parte da direita são apresentadas as informações que constituem a tarefa selecionada (conforme mencionado no requisito 3).

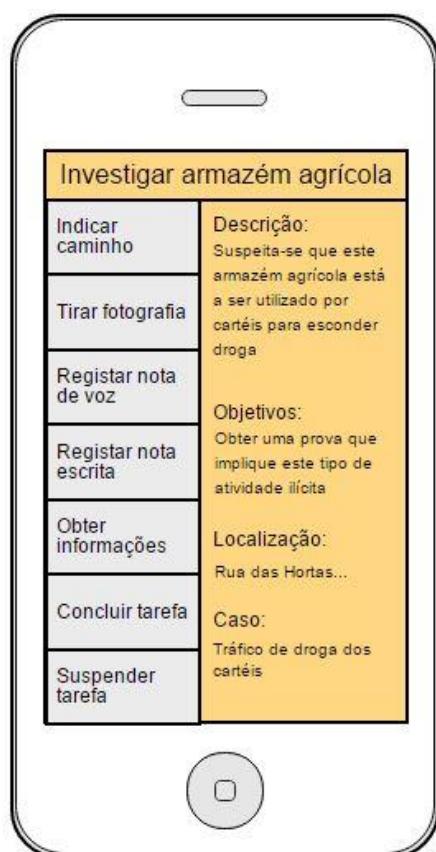


Figura 41 Mockup Realizar Tarefa

De seguida, é apresentado o *mockup* correspondente à obtenção de informação complementar. Este consiste numa caixa de texto em que o utilizador insere a palavra ou expressão que pretende pesquisar na *internet* (de maneira a obter informações sobre esse dado assunto) com o auxílio de um motor de busca. Quando a pesquisa é efetuada, os primeiros resultados são apresentados na zona abaixo da caixa de texto.



Figura 42 Mockup Obter informações complementares

6. Implementação

6.1. BackOffice

6.1.1. Namespaces

A implementação do *BackOffice* iniciou-se pela divisão do projeto em dois projetos distintos (ou *namespaces*). Um deles denominado de *BackOffice* e o outro de *Data*.

Esta divisão surge na necessidade de ter o software desenvolvido numa organização de três níveis na arquitetura principal do sistema de *BackOffice*: camada de apresentação, camada de negócio e camada de dados.

Porém, tal como acabado de referir, o projeto de sistema *BackOffice* apenas foi dividido em dois projetos, isto porque a camada de apresentação e a camada de negócio estão juntas. O projeto *BackOffice* tem inserido a camada de negócio através de todas as funções e regras de todo o negócio, corresponde à implementação dos diversos métodos. Este projeto tem também inserido a camada de apresentação através das janelas implementadas para cada funcionalidade.

Relativamente à camada de dados, esta está inserida no projeto denominado de *Data*, no qual contem os *Entity Data Models* que servem para permitir a conexão com a base de dados. Esta conexão à base de dados será explicada na seguinte secção.

6.1.2. Conexão à base de dados (*EntityFramework*)

A conexão com a base de dados é feita através de *Entity Data Models*, desta forma foram criados dois *Entity Data Models*: *ModelDiretor* e *ModelIC*. Para cada modelo é criada uma conexão com a base de dados, nesta conexão é usado o *login* respetivo a Diretor e Inspetor-Chefe. Por exemplo, para a criação do *ModelDiretor* foi criado uma conexão com a base de dados com o *login* de Diretor, sendo importadas para o modelo as tabelas e *views* associadas ao *login*. Para a criação do *ModeloIC* o processo foi semelhante, foi criado uma conexão com a base de dados com o *login* IC, sendo importadas para o modelo as tabelas, *views* e *Stored Procedure* associados ao *login*.

De forma a explicar a forma como se procede a comunicação com a base de dados vamos apresentar a implementação do *login*. Inicialmente é apresentado o menu que permite aos utilizadores do sistema se autenticarem indicando o seu id e password. Com esta informação, procede-se à validação das credenciais, esta validação tem de ser feita com base na informação existente na base de dados. É então instanciado um objeto do tipo *EntitiesDiretor*

que servirá de ponto da aplicação com a base de dados. Com base nesta variável criada do tipo *EntitiesDiretor*, caso não exista nenhum utilizador com o id indicado é mostrada uma mensagem de erro a indicar a não existência do utilizador. Esta verificação é feita da seguinte forma:

```
if (!dados.Utilizador.Any(u => u.IdUtilizador == id))
{
    MessageBox.Show("O utilizador não existe!", "Atenção", MessageBoxButton.OK, MessageBoxIcon.Error);
    txtLogIN.Focus();
    return;
}
```

Figura 43 - Utilização de um Entity Data Model

Posteriormente é feita uma nova verificação, desta vez incluindo a verificação da validade da *password*. Caso a *password* esteja incorreta é mostrada uma imagem de erro a indicar a que password está incorreta. Caso não ocorra nenhum destes casos, significa que os dados são válidos, sendo deste modo permitido ao utilizador usufruir das restantes funcionalidades consoante o seu tipo de utilizador. De seguida é apresentado o código na integra relativo ao *login*, no qual é evidenciado a forma como é feita a comunicação com a base de dados:

```
private void button_Click(object sender, RoutedEventArgs e)
{
    try
    {
        int id = Convert.ToInt32(txtLogIN.Text);
        using (var dados = new Data.EntitiesDiretor())
        {
            if (!dados.Utilizador.Any(u => u.IdUtilizador == id))
            {
                MessageBox.Show("O utilizador não existe!", "Atenção", MessageBoxButton.OK, MessageBoxIcon.Error);
                txtLogIN.Focus();
                return;
            }
            if (!dados.Utilizador.Any(u => u.IdUtilizador == id && u.Password == txtPass.Password))
            {
                MessageBox.Show("A password está incorreta", "Atenção", MessageBoxButton.OK, MessageBoxIcon.Error);
                txtPass.Focus();
            }
            else {
                int tipoUtilizador = dados.Utilizador.First(u => u.IdUtilizador == id).IdTipoUtilizador;
                Menu menu = new Menu(id, tipoUtilizador);
                this.Close();
                menu.Show();
            }
        }
    catch (FormatException)
    {
        MessageBox.Show("Tem que inserir o ID", "Atenção", MessageBoxButton.OK, MessageBoxIcon.Error);
        return;
    }
}
```

Figura 44 - Código *login*

Outra abordagem para conectar o *BackOffice* à base de dados seria apenas criar um *Entity Data Model*. Este *Entity Data Model* faria a conexão à base de dados com o perfil de administrador e a permissão das funcionalidades aos diferentes tipos de utilizador apenas seria feita através do *BackOffice*. Com a nossa implementação (criar dois *Entity Data Models* com o acesso de cada utilizador) assegura a permissão tanto a nível aplicacional como no SGBD. Esta

solução seria impraticável no caso de existirem vários utilizadores (pois seria necessário a criação de vários *Entity Data Models*). No entanto, como este não é o caso preferimos implementar a solução que acrescenta mais segurança.

6.1.3. Interface

A interface gráfica da aplicação para o *BackOffice* permite a interação entre os utilizadores do sistema (neste caso, apenas Diretor e Inspetores-Chefe) e a aplicação criada. Para a conceção das diversas janelas foram utilizados inúmeros componentes de forma a ser possível a realização das funcionalidades estipuladas.

A primeira janela criada é relativa ao *login* que os usuários têm de efetuar para poderem usufruir das funcionalidades que lhes são inerentes, estas funcionalidades são distintas consoante o tipo de utilizador. As componentes utilizadas para esta ação são: *TextBox* para permitir ao utilizador inserir o seu id e a sua *password*, *Button* para permitir confirmar as credenciais. São usadas *Labels* para indicar o que se pretende que seja escrito em cada *TextBox*. A seguinte figura é referente à janela de *login*:

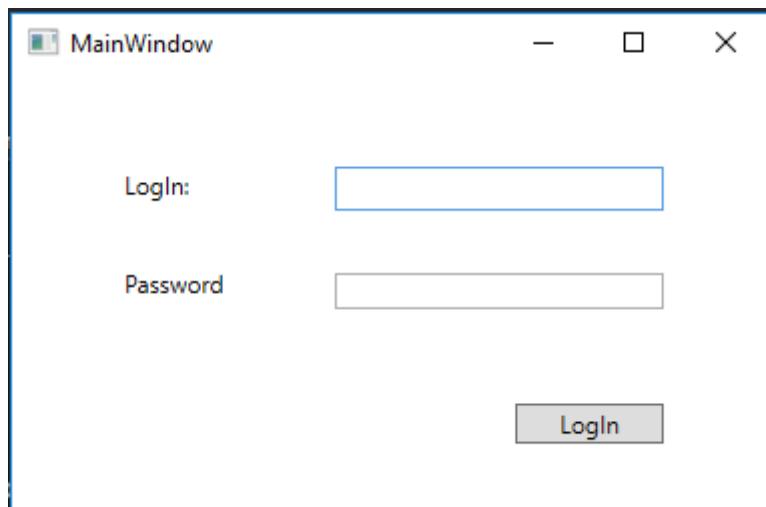


Figura 45 - Janela *login*

Após ter sido efetuado o *login* é apresentado ao utilizador uma janela menu no qual tem identificado todas as funcionalidades que o utilizador pode efetuar no *BackOffice*. Tal como referido anteriormente, os utilizadores do *BackOffice* são apenas o Diretor e os Inspetores-Chefe, deste modo, este menu apresentado difere nas funcionalidades apresentadas consoante o tipo de utilizador. Esta verificação é feita aquando do *login*, sendo verificado o tipo de utilizador e apresentando, desta forma, um menu com possibilidades de escolha diferentes.

A componente usada para apresentar as funcionalidades e possibilitar a escolha da pretendida é uma *ListBox*. Foi escolhido esta componente uma vez que desta forma é possível apresentar todas as funcionalidades e é permitido escolher uma delas clicando sobre a mesma. As seguintes figuras representam os dois menus possíveis, ou seja, caso o utilizador seja um Diretor e caso seja um Inspetor-Chefe:

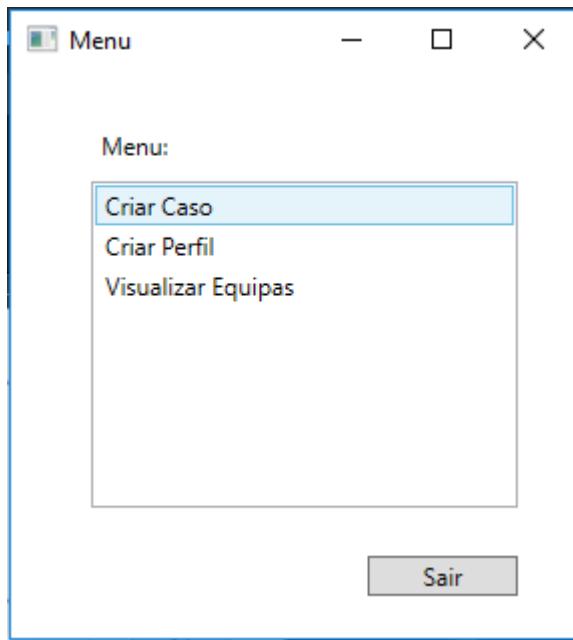


Figura 46 - Menu Diretor

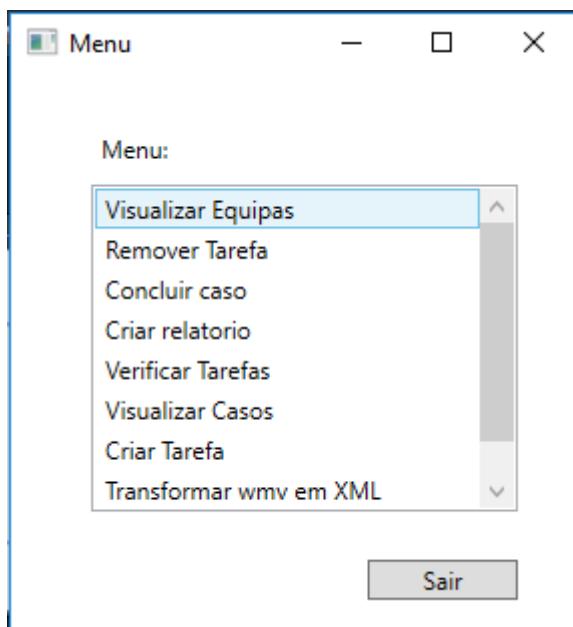


Figura 47 - Menu IC

Qualquer opção selecionada num dos menus acabados de apresentar leva à mostragem da janela correspondente. Por exemplo, caso o utilizador logado seja do tipo Inspetor-Chefe, este tem a possibilidade de criar tarefas. Assim, caso seja essa a opção escolhida, é apresentada a janela correspondente, no qual existe os vários campos essenciais à criação de uma tarefa. Esta informação que é necessária foi estipulada anteriormente na fundamentação e especificação do trabalho. Para permitir ao utilizador inserir o título, descrição, objetivos e o local de investigação da tarefa são usadas *TextBox*, visto permitirem a inserção de texto. Para escolher o caso ao qual a tarefa esta inserida e o agente responsável pela mesma, são usadas *ComboBox*, visto mostrarem apenas os casos e agentes possíveis e selecionar diretamente sem a necessidade de escrita. Existe também um botão que permite ver o estado dos casos referentes ao Inspetor-Chefe logado, um botão de confirmação de criação de tarefa e um último que permite fechar a janela. A seguinte figura representa a janela criada para a funcionalidade de criar uma tarefa:

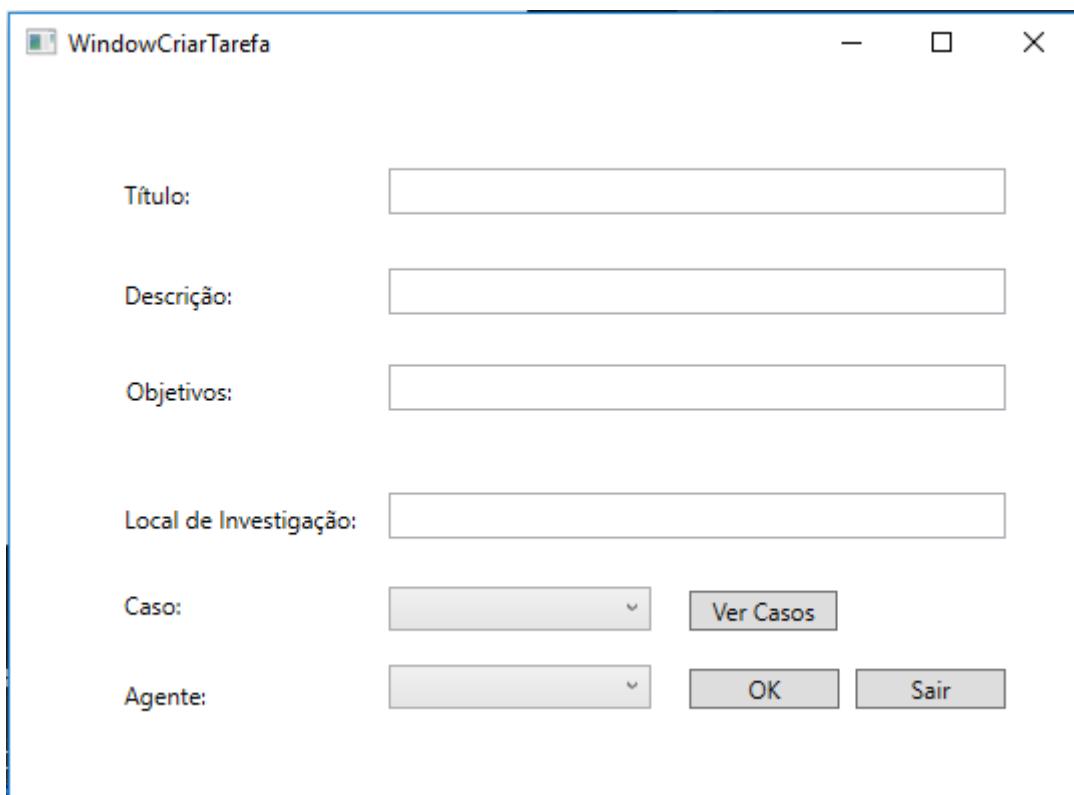


Figura 48 - Janela referente a criar uma tarefa

Por forma a demonstrar mais algumas janelas da interface gráfica, serão apresentadas de seguida algumas janelas criadas referentes a diversas funcionalidades:

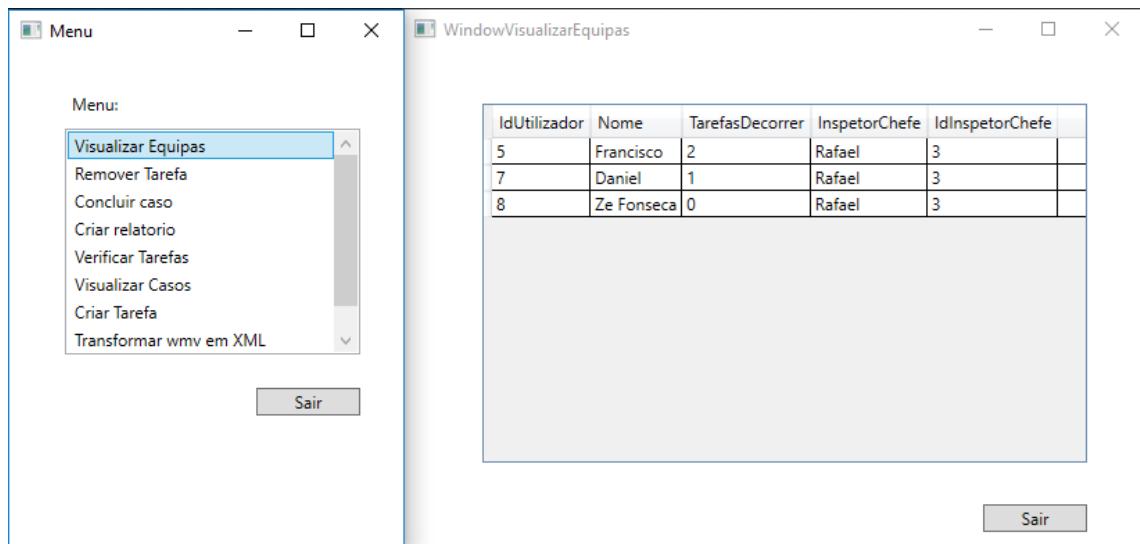


Figura 49 - Janela Menu e janela visualizar equipas

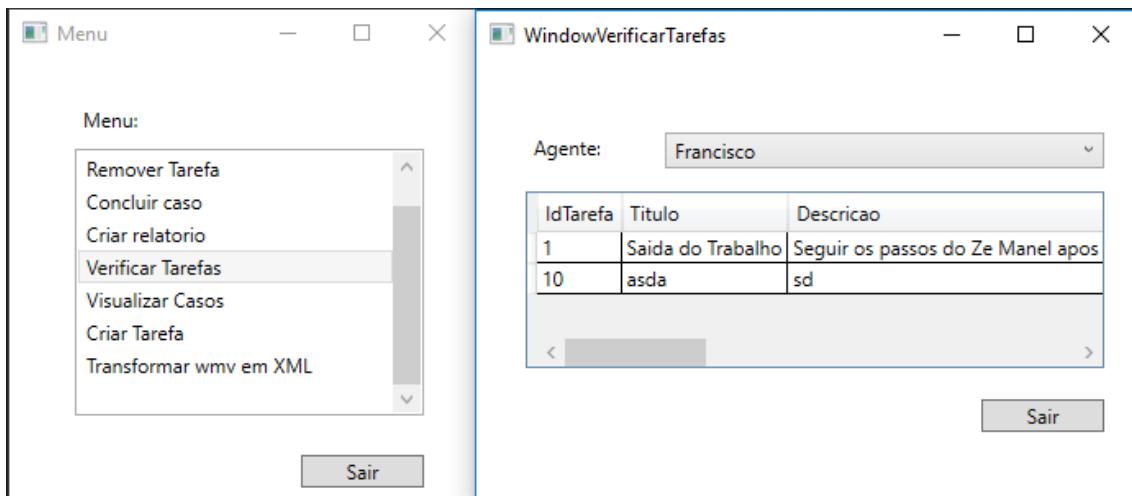


Figura 50 - Janela Menu e janela verificar tarefas

Em relação às componentes utilizadas, muitas foram indicadas ao longo desta secção, porém, foram utilizadas outras componentes que ainda não foram mencionadas.

Exemplificando, foi utilizada a componente *RadioButton* para permitir a um Diretor escolher o tipo de utilizador na criação de um perfil. Na seguinte figura é ilustrado esta particularidade da janela de criação de perfil:

Tipo de perfil: Inspector-Chefe Agente

Figura 51 - Utilização de *RadioButton*

Outra componente várias vezes utilizada na amostragem de dados da base de dados em forma de tabela foi a componente *DataGrid*. Funcionalidades como visualizar casos ou tarefas, a forma como estes dados são apresentados aos utilizadores é através desta componente. De seguida é apresentado um exemplo no qual o utilizador visualiza dados através de um *DataGrid*:

Casos:

IdCaso	Nome	Descricao
1	Investigacao ZeManel	A esposa do Zé Manel pensa que o marido a an Posto isto, foi-nos pedido a investigação desta s

Figura 52 - Exemplo *DataGrid*

A interface do sistema de BackOffice não segue estritamente os *Mockups* definidos. Alguns são seguidos quase na totalidade, no entanto existe casos em que se diferenciam um pouco do planeado. A razão prende-se essencialmente pelo facto de que quando foram criados os *mockups* não tínhamos uma boa noção de quais *WPF Controls* existiam. Outro factor foi facilitar o trabalho a desenvolver, atingindo o mesmo objetivo. Exemplo disso é o menu: o menu foi implementado usando uma *ListBox* no entanto existem outras maneiras de implementar o pretendido e, possivelmente, mais atrativas visualmente. No entanto, não achamos que isso fosse algo prioritário e decidimos “simplificar” e aplicar o tempo a cumprir os requisitos/funcionalidades.

6.1.4. Funcionalidades

O sistema de *BackOffice* tem inserido uma grande variedade de funcionalidades que os seus utilizadores podem usufruir. Como já referido anteriormente, o sistema de *BackOffice* apenas é usado para utilizadores do tipo Diretor ou Inspetores-Chefe, tendo diferentes funcionalidades possíveis consoante este tipo.

As funcionalidades implementadas para utilizadores do tipo Diretor foram: criar caso, criar perfil e visualizar equipas. Tendo em conta os requisitos estas são as funcionalidades permitidos aos utilizadores do tipo Diretor.

Por sua vez, Inspetores-Chefe têm um maior número de funcionalidades associadas. Sendo essas funcionalidades as seguintes: visualizar equipas, remover tarefas, concluir caso, criar relatório, verificar casos, criar tarefas, transformar ficheiros áudio .wav em documentos anotados XML.

De forma a melhor descrever o processo de implementação destas funcionalidades, apresenta-se de seguida algumas implementações em concreto de funcionalidades.

6.1.4.1. Criar tarefa

A criação de uma tarefa requer que sejam inseridas diversas informações, apenas é criada uma tarefa se for inserido o título que se pretende dar à tarefa, uma descrição, os objetivos a alcançar, o local de investigação, o caso a que é referente e o agente que será responsável pela realização da tarefa.

Inicialmente é necessário preencher as *ComboBox* referentes à escolha de caso e agente. A atribuição de uma tarefa apenas pode ser feita a um caso que seja associado ao Inspetor-Chefe, desta forma, é feita uma conexão à BD e através do *Stored Procedure* CasosAtivos é buscada a informação dos casos que são associados ao IC e que ainda não tenham sido terminados. O que é mostrado ao utilizador na interface é apenas o nome dos casos possíveis, sendo o valor do item selecionado na *ComboBox* a nível de implementação o IdCaso. O processo para a escolha do Agente é semelhante ao anterior, apenas são mostrados agentes pertencentes às equipas do IC. Na figura seguinte é demonstrado esta implementação:

```
private Data.EntitiesIC data = new Data.EntitiesIC();
private int idIC;
public CriarTarefa(int idIC)
{
    this.idIC = idIC;
    InitializeComponent();
    var listaCasos = (from c in data.CasosAtivos(this.idIC)
                      where c.Terminado == false
                      select new { c.IdCaso, c.Nome, c.Descricao, c.Objetivos, c.Relatorio, c.Terminado }).ToList();
    comboBoxCaso.ItemsSource = listaCasos;
    comboBoxCaso.DisplayMemberPath = "Nome";
    comboBoxCaso.SelectedValuePath = "IdCaso";

    var listaAgentes = (from t in data.Utilizador
                        where t.IdChefe == this.idIC
                        select new { t.IdUtilizador, t.IdTipoutilizador, t.Nome, t.Password, t.IdChefe }).ToList();
    comboBoxAgente.ItemsSource = listaAgentes;
    comboBoxAgente.DisplayMemberPath = "Nome";
    comboBoxAgente.SelectedValuePath = "IdUtilizador";
}
```

Figura 53 - Tratamento *ComboBox* (*Criar tarefa*)

Um Inspetor-Chefe pode confirmar a criação da tarefa ao clicar em *OK*, sendo feita uma série de verificações a nível de código para permitir a criação da tarefa. A primeira é verificar se todos os campos estão preenchidos, em caso negativo é mostrada uma mensagem de erro. Caso todos os campos estejam preenchidos, é realizada uma conversão do local de investigação indicado nas respetivas coordenadas geográficas. De seguida é criada uma nova tarefa com os dados indicados pelo utilizador, sendo esta nova tarefa inserida na base de dados. Na seguinte figura é apresentado o código referente a esta implementação:

```

private void buttonOK_Click(object sender, RoutedEventArgs e)
{
    if(textBoxTitulo.Text != "" && textBoxDescricao.Text != "" && textBoxObjetivos.Text != "" && textBoxLocal.Text != "" && comboBoxCaso.SelectedIndex > -1 && comboBoxAgente.SelectedIndex > -1)
    {
        string url = "http://maps.googleapis.com/maps/api/geocode/json?sensor=true&address=";
        Data.Tarefa novaTarefa = new Data.Tarefa();
        dynamic googleResults = new Uri(url + textBoxLocal.Text).GetDynamicJsonObject();
        foreach (var result in googleResults.results)
        {
            Console.WriteLine("[" + result.geometry.location.lat + "," + result.geometry.location.lng + "] " + result.formatted_address);
            novaTarefa.Latitude = (double)result.geometry.location.lat;
            novaTarefa.Longitude = (double)result.geometry.location.lng;
        }

        novaTarefa.Titulo = textBoxTitulo.Text;
        novaTarefa.Descrição = textBoxDescricao.Text;
        novaTarefa.Objetivos = textBoxObjetivos.Text;
        novaTarefa.Realizada = false;
        novaTarefa.Exportada = false;
        novaTarefa.IdCaso = Convert.ToInt32(comboBoxCaso.SelectedValue);
        novaTarefa.IdAgente = Convert.ToInt32(comboBoxAgente.SelectedValue);
        data.Tarefa.Add(novaTarefa);
        data.SaveChanges();
        MessageBox.Show("Tarefa criada");
    }
    else
    {
        MessageBox.Show("Campos inválidos!", "Atenção", MessageBoxButton.OK, MessageBoxImage.Error);
    }
}

```

Figura 54 - Código referente à criação de uma tarefa

6.1.4.2. Enviar Email

Enviar *emails* dos relatórios gerados é uma das funcionalidades que o sistema de *BackOffice* permite realizar. Quando são gerados relatórios de casos que tenham sido terminados e ainda não tenham sido criados os relatórios, é possível usufruir desta funcionalidade.

O processo de envio de relatório passa pela indicação do *email* e escolha do relatório. Foi definido uma mensagem base para o corpo do *email*, no qual é enviado o anexo referente ao ficheiro *.pdf*. Foi também definido o assunto do *email*. O envio do *email* é feito com o auxílio da classe *TEmail* no qual é definido o *email* origem de envio, e criadas condições ao envio. Na seguinte figura é apresentado a classe *TEmail* que permite o envio de *emails*:

```

public class TEmail
{
    private System.Net.Mail.MailMessage Mensagem;
    private System.Net.Mail.SmtpClient Servidor;

    private const string EMAIL_SMTP = "smtp.gmail.com";
    private int EMAIL_PORT = 587;
    private int EMAIL_TIMEOUT = 10000;
    private string EMAIL_PASSWORD = "raposaseca";
    private string EMAIL_CONTA = "armindoquilhoado@gmail.com";

    public bool Enviar(string Endereco, string Assunto, string Texto, string caminhoRelatorio)
    {
        try
        {

            Servidor = new System.Net.Mail.SmtpClient(EMAIL_SMTP, EMAIL_PORT);
            Servidor.Timeout = EMAIL_TIMEOUT;

            System.Net.NetworkCredential Credencial = new System.Net.NetworkCredential(EMAIL_CONTA, EMAIL_PASSWORD);
            Servidor.UseDefaultCredentials = false;
            Servidor.Credentials = Credencial;
            Servidor.EnableSsl = true;
            Servidor.Timeout = 50000;

            Mensagem = new System.Net.Mail.MailMessage();
            Mensagem.To.Add(Endereco);
            Mensagem.Subject = Assunto;
            Mensagem.From = new System.Net.Mail.MailAddress(EMAIL_CONTA);
            Mensagem.Body = Texto;
            Mensagem.Attachments.Add(new System.Net.Mail.Attachment(caminhoRelatorio));

            Servidor.Send(Mensagem);
            Mensagem.Dispose();
            Servidor.Dispose();
            MessageBox.Show("Email enviado com sucesso");
            return true;
        }
    }
}

```

Figura 55 - Classe TEmail

6.1.4.3. Transformar voz em xml

O processo de transformação de ficheiro áudio em documentos anotados XML inicia-se pela definição das palavras-chave. Ou seja, existe uma serie de palavras-chave que os agentes têm de conhecer para gravarem o ficheiro de voz de forma correta.

As palavras-chave escolhidas são em inglês, uma vez ser mais ter sido o solicitado pelo cliente. O conjunto das palavras-chaves são: “Terminate”, “Attention”, “Clothing”, “Danger”, “Status” e “Crime”. A palavra “Terminate” indica o fim de gravação, sendo que as palavras ditas após esta palavra não serão armazenadas. Na seguinte figura é apresentado a definição das palavras-chave:

```

private void inicializarPalavrasChave()
{
    palavrasChave = new List<String>();
    palavrasChave.Add("Terminate");
    palavrasChave.Add("Attention");
    palavrasChave.Add("Clothing");
    palavrasChave.Add("Danger");
    palavrasChave.Add("Status");
    palavrasChave.Add("Crime");
}

```

Figura 56 - Definição das palavras-chave

De seguida são apresentadas ao utilizador as tarefas dos agentes pertencentes ao IC que contenham dados do tipo ficheiro de áudio.

Posteriormente, ao confirmar a intenção de transformação é armazenado numa *List<String>* todas as palavras ditas na gravação pelo agente de forma ordenada. Na seguinte figura é apresentado o reconhecimento das palavras do ficheiro áudio:

```

private List<String> palavras(string caminho)
{
    List<String> palavras = new List<string>();

    sre.LoadGrammar(gr);
    sre.SetInputToWaveFile(caminho);
    sre.BabbleTimeout = new TimeSpan(Int32.MaxValue);
    sre.InitialSilenceTimeout = new TimeSpan(Int32.MaxValue);
    sre.EndSilenceTimeout = new TimeSpan(100000000);
    sre.EndSilenceTimeoutAmbiguous = new TimeSpan(100000000);

    while (true)
    {
        try
        {
            var recText = sre.Recognize();
            if (recText == null)
            {
                break;
            }

            palavras.Add(recText.Text);
        }
        catch (Exception ex)
        {
            MessageBox.Show("Erro!");
            break;
        }
    }
    return palavras;
}

```

Figura 57 - Reconhecimento das palavras do ficheiro audio

Após a geração da lista é criado um documento *XML*, no qual, à medida que vão sendo encontradas palavras-chaves é guardada a informação como valores de atributos. Quando for

encontrada a palavra-chave “Terminate” todas as restantes são descartadas. De seguida apresentamos a implementação da criação do documento anotado (*XML*):

```
bool gerarXML(List < String > palavras)
{
    bool gerou = false;

    XmlDocument doc = new XmlDocument();
    XmlDeclaration xmlDeclaration = doc.CreateXmlDeclaration("1.0", "UTF-8", null);
    XmlElement root = doc.DocumentElement;
    doc.InsertBefore(xmlDeclaration, root);

    XmlElement element = doc.CreateElement(string.Empty, "body", string.Empty);
    doc.AppendChild(element);

    string palavraC = "", frase = "";
    bool first = true;
    XmlElement element1 = null, element2 = null;
   XmlAttribute atributo = null;
    foreach (var palavra in palavras)
    {

        if (this.palavrasChave.Any(x => x.Equals(palavra)) && !palavraC.Equals(palavra) && first == true)
        {//é primeira palavra chave a aparecer
            first = false;
            frase = "";
            palavraC = palavra;
            element1 = doc.CreateElement(string.Empty, "pr", string.Empty);
            atributo = doc.CreateAttribute("pal");
            atributo.Value = palavra;
            element1.Attributes.Append(atributo);
            element.AppendChild(element1);
        }

        else if (palavra.Equals("Terminate") && first==false)
        { //palavra reservada de fecho
            palavraC = palavra;
            element2 = doc.CreateElement(string.Empty, "texto", string.Empty);
            atributo = doc.CreateAttribute("txt");
            atributo.Value = frase;
            element2.Attributes.Append(atributo);
            element1.AppendChild(element2); //element2 filho de element1

            frase = "";
            gerou = true;
            break;
        }

        else if (this.palavrasChave.Any(x => x.Equals(palavra)) && !palavraC.Equals(palavra) )
        { //Palavra chave no meio da sequência
            palavraC = palavra;
            element2 = doc.CreateElement(string.Empty, "texto", string.Empty);
            atributo = doc.CreateAttribute("txt");
            atributo.Value = frase;
            element2.Attributes.Append(atributo);
            element1.AppendChild(element2); //element1 filho de element
            frase = "";

            //criar novo elemento
            element1 = doc.CreateElement(string.Empty, "pr", string.Empty);
            atributo = doc.CreateAttribute("pal");
            atributo.Value = palavra;
            element1.Attributes.Append(atributo);
            element.AppendChild(element1);
        }

        else //palavras que não são reservadas
        {
            frase += palavra + " ";
        }
    }
}
```

Figura 58 - Implementação da criação de documento *XML*

Após a criação de um documento anotado, procede-se à remoção do ficheiro áudio e à mudança do atributo Caminho na tabela Dados referente ao dado em questão. De seguida é apresentado um exemplo de documento anotado criado para a seguinte sequência de palavras “Attention john is seen with his mistress Danger the john's wife is near” :

```
<?xml version="1.0" encoding="UTF-8"?>
<body>
    <pr pal="Attention">
        <texto txt="john is seen with his mistress " />
    </pr>
    <pr pal="Danger">
        <texto txt="the john's wife is near " />
    </pr>
</body>
```

Figura 59 - Exemplo XML

6.2. *FrontOffice*

O *FrontOffice* constitui a aplicação que os Agentes utilizam nos seus dispositivos móveis durante a realização do trabalho de campo. Esta aplicação permite-lhes ter acesso às tarefas que devem realizar, bem como às funcionalidades de auxílio para a sua investigação

Nesta secção vão ser detalhados os procedimentos seguidos para a implementação deste subsistema, descrevendo as funcionalidades e serviços que o *FrontOffice* oferece. Além disso, será também especificado de que maneira este subsistema oferece essas funcionalidades, de modo a ir de encontro ao especificado nos requisitos validados com o cliente e, consequente, àquilo que foi traduzido nos diagramas UML.

6.2.1. Ferramentas utilizadas

Nesta secção vão ser detalhadas as ferramentas que o grupo de desenvolvimento utilizou para desenvolver a aplicação móvel de *FrontOffice*, bem como as razões que levaram a adotar essas ferramentas para realizar o projeto.

6.2.1.1. Visual Studio 2015

O uso deste poderoso IDE no desenvolvimento deste projeto deveu-se ao facto de ser um IDE poderoso que possui múltiplas ferramentas e funcionalidades que ajudam a desenvolver uma aplicação de forma mais segura e que pode ser adaptada a varias plataformas. Isso traz como acréscimo melhorias na produtividade da equipa e na gestão do ciclo de vida da aplicação e permite a utilização das mais recentes tecnologias.

6.2.1.2. Android

O grupo decidiu desenvolver a aplicação focalizada para este sistema operativo por ser o sistema operativo para dispositivos moveis mais utilizado no mundo. Além disso, o cliente sugeriu que fosse esse o foco da nossa aplicação, uma vez que a maioria dos seus agentes possuíam dispositivos que continham este sistema operativo.

6.2.1.3. Xamarin

O grupo decidiu adotar o uso desta plataforma tendo em conta que permite desenvolver aplicações para várias plataformas(IOS, Android, Windows Phone) usando a mesma linguagem, C#, que foi a linguagem que o cliente pediu para o desenvolvimento do projeto. Além disso, possui ferramentas de IDE que ajudam a aumentar a produtividade da produção de código.

6.2.2. Menu Principal

No momento do arranque da aplicação, ao utilizador do *FrontOffice* é lhe apresentado um pequeno menu, com 4 botões que representam 4 diferentes escolhas que o utilizador pode tomar. Cada uma destas escolhas generaliza um conjunto de procedimentos, todos eles distintos, sendo que a intencionalidade dessas escolhas é refletida na informação colocada no interior dos botões. O utilizador poderá então escolher entre ver tarefas, iniciar importação, iniciar exportação e alterar credenciais, através de um clique no respetivo botão.

Caso escolha a primeira alternativa, o utilizador(agente) poderá consultar todas as tarefas que possui no seu dispositivo bem como alterar o seu estado e adicionar-lhe diferentes tipos de dados. A segunda e terceira escolha permitem comunicar o *BackOffice* e transferir/atualizar dados, enquanto que a quarta opção permite ao agente alterar as suas credencias de acesso.

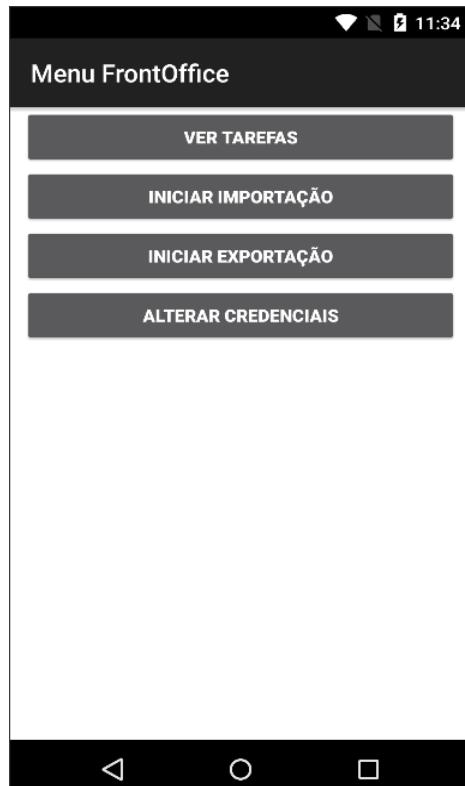


Figura 60 Menu principal *FrontOffice*

6.2.3. Persistência de dados

A persistência de dados neste subsistema é garantida através de um ficheiro XML, localizado numa diretoria localizada no sistema de ficheiros do dispositivo, onde é guardada toda a informação necessária não só para o agente realizar o seu trabalho de campo, como também para manter a sua informação organizada e para comunicar com o *BackOffice*.

O uso de um ficheiro XML para representar e armazenar a informação contida neste subsistema tem o objetivo de poupar espaço no dispositivo do agente, uma vez que a outra alternativa seria incluir uma base de dados no dispositivo, o que levaria a que a aplicação ocupasse mais espaço. Quando o agente altera o estado de uma tarefa ou lhe associa um determinado tipo de dado, essas alterações são refletidas no ficheiro XML, após serem efetuadas.

Na imagem seguinte é possível visualizar um exemplo de um ficheiro XML, estruturado da mesma forma que o ficheiro contido no dispositivo móvel.

```

<?xml version="1.0" encoding="utf-8" ?>
<PlanoAtividades IdAgente="Jose Paulo" Password="gatinhos123">

    <Tarefa IdTarefa="1" NomeCaso="Roubo em Braga" IdCaso="2" Titulo="Espiar Carlos" Realizada="false" Suspensa="false" Latitude="41,561937" Longitude="-8,396994" >
        <LocalizacaoTexto>Rua da Universidade, Guimarães</LocalizacaoTexto>
        <Descricao> Ir atrás dele </Descricao>
        <Objetivos>
            <Objetivo> Apanha-lo a roubar o café </Objetivo>
            <Objetivo> Apanha-lo a roubar a loja </Objetivo>
        </Objetivos>
        <Dados>
            <Imagen caminho = "\fotos\img1.png" Latitude="41" Longitude="-8.396994" Data="2014-11-04 11:45:34"/>
            <Gravacao caminho = "\gravacoes\grav2" Latitude="41.561937" Longitude="54.37" Data ="2014-11-04 11:45:34"/>
            <NotaEscrita Latitude="22" Longitude="15" Data ="2014-11-04 11:45:34">
                Ele foi lá, mas só roubou guardanapos
            </NotaEscrita>
        </Dados>
    </Tarefa>
    <Tarefa IdTarefa="2" NomeCaso="Monte na Barca" IdCaso="3" Titulo="Recolher provas incriminadoras" Realizada="false" Suspensa="false" Latitude="41.561937" Longitude="-8.396994" >
        <LocalizacaoTexto>Monte do Rafa, Barca</LocalizacaoTexto>
        <Descricao> Arranjar provas para incriminar o Malaquias </Descricao>
        <Objetivos>
            <Objetivo> Recolher todas as provas possíveis </Objetivo>
        </Objetivos>
        <Dados>
            <Imagen caminho = "\fotos\img1.png" Latitude="41.561937" Longitude="54.37" Data ="2014-11-04 11:45:34"/>
            <Gravacao caminho = "\gravacoes\grav2" Latitude="41.561937" Longitude="54.37" Data ="2014-11-04 11:45:34"/>
            <NotaEscrita Latitude="22" Longitude="11" Data ="2014-11-04 11:45:34">
                </NotaEscrita>
        </Dados>
    </Tarefa>
</PlanoAtividades>

```

Figura 61 Exemplo de um ficheiro XML que representa o estado do FrontOffice

Como é observável na imagem anterior, a raiz do ficheiro XML é o plano de atividades, que refere um agente (através do seu ID) e a sua palavra-chave. Dentro da raiz surgem várias tarefas, que são associadas segundo o que foi especificado no requisito funcional 3.1, tendo como atributos o seu ID, o título da tarefa, o nome e identificação do caso a que está associada, o local sob a forma de coordenadas geográficas (latitude e longitude) bem como a informação sobre o seu estado (realizada e suspensa). Além disso dentro de cada tarefa surgem os seus objetivos, a descrição e todos os dados a ela associados, sendo diferenciados pelo seu tipo (nota escrita, imagem ou gravação de voz) e tendo como atributos a latitude e longitude (cumprindo assim o requisito de utilizador 17) do local onde foram recolhidos, bem como o seu caminho no sistema de ficheiros e data do seu registo.

6.2.4. Ver tarefas

Quando o utilizador prime o botão de Ver Tarefas, tem acesso à lista de tarefas que se encontra disponível no seu dispositivo. Nessa lista são apresentadas as tarefas que lhe estão atribuídas, sendo exibido para cada uma a identificação do caso, o título da tarefa e também a sua identificação.

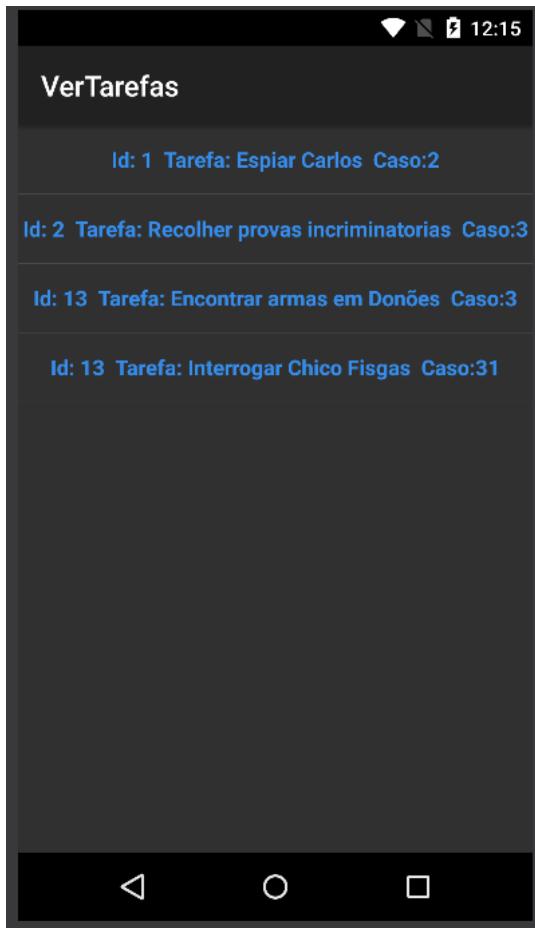


Figura 62 Ver tarefas

Através de um clique sobre a tarefa pretendida, o agente tem acesso a um menu onde será apresentada informação da tarefa com mais detalhe e serão apresentadas outras opções para associar dados à tarefa e/ou alterar o seu estado.

6.2.5. Importação de tarefas

No menu principal, é apresentado ao agente um botão que realiza a importação de tarefas a partir do *BackOffice*. Esta funcionalidade permite ir de encontro ao que foi especificado nos requisitos (requisito funcional de sistema 13.1) e consiste na importação de tarefas atribuídas no *BackOffice* pelo Inspetor Chefe e que são destinadas ao agente portador do dispositivo móvel (requisito funcional de sistema 13.2).

Para a concretização desta funcionalidade, o agente tem que estar ligado à mesma rede local que o servidor de *BackOffice*. O *FrontOffice* tentará após o clique do botão estabelecer

ligação com o *BackOffice*, enviando as suas credencias de acesso. Caso a conexão seja estabelecida, e as credenciais sejam válidas, é construída uma *query* em SQL na aplicação móvel que solicita todas as tarefas atribuídas a esse agente que ainda não foram exportadas. O servidor do *BackOffice* enviará as respetivas tarefas, sendo que após a receção destas o *FrontOffice* irá enviar uma nova instrução ao servidor para que atualize as tarefas enviadas, marcando-as como exportadas. Após estes procedimentos a conexão é fechada e cada tarefa importada e respetiva informação é gravada no ficheiro XML.

```

try
{
    IDbConnection condb;
    using (condb = new SqlConnection(connectionString)) {
        condb.Open();
        using(IDbCommand cmddb = condb.CreateCommand())
        {
            string sql = "SELECT * FROM [L14-Agentes].[dbo].[Utilizador] WHERE IdUtilizador =" + idAgente;

            cmddb.CommandText = sql;
            using (IDataReader reader = cmddb.ExecuteReader())
            {
                string palavraChave = null;
                while (reader.Read())
                {
                    palavraChave = (string)reader["Password"];
                }
            }
            if (!palavraChave.Equals(pass))
            {
                var callDialog = new AlertDialog.Builder(this);
                callDialog.SetMessage("Credenciais inválidas");
                callDialog.SetNegativeButton("Ok", delegate {
                });
                callDialog.Show();
                return;
            }
        }
        reader.Close();
        sql = "SELECT * FROM [L14-Agentes].[dbo].[Tarefa] as T INNER JOIN [L14-Agentes].[dbo].[Caso] as C ON T.IdCaso=C.IdCaso WHERE IdAgente=" + idAgente + " AND Exportada=0";
        cmddb.CommandText = sql;
        IDataReader reader3 = cmddb.ExecuteReader();
}

```

Figura 63 Excerto de código que permite estabelecer conexão ao BackOffice

6.2.6. Exportação de tarefas

Com vista a cumprir o que foi pedido pelo cliente e cumprir o que foi convenientemente especificado nos requisitos e modelado nos diagramas UML, o *FrontOffice* oferece a possibilidade de exportar as tarefas e dados associados para o *BackOffice*. Conforme especificado nos requisitos funcionais de sistema 19.1 e 19.2, apenas as tarefas marcadas pelo agente como concluídas serão exportadas, sendo que as restantes tarefas permanecem no dispositivo até serem concluídas (requisito funcional de sistema 19.3). Quando o utilizador(agente) premir o terceiro botão do menu principal, o botão “Iniciar Exportação”, o processo de exportação inicia-se. O ficheiro XML é consultado, sendo recolhidas todas as tarefas que foram marcadas como concluídas, bem como toda a informação e dados inerentes a elas. De seguida, tal como na importação, é estabelecida ligação com o *BackOffice*, sendo enviadas as credenciais de acesso do agente. Caso a conexão seja estabelecida, e as credenciais sejam

válidas, é construída uma *query* em SQL na aplicação móvel que vai fazer com que todas as tarefas que vão ser exportadas sejam marcadas na BD do *BackOffice* como concluídas, e de seguida uma outra que vai enviar todos os dados (registos de voz, notas escritas e fotografias) referentes às tarefas que vão ser enviadas. Por fim, são removidas todas as tarefas enviadas no ficheiro XML do dispositivo móvel (conforme especificado no requisito funcional de sistema 21.3).

```

string connectionString = "Server=192.168.111.110;Database=LI4-Agentes;User ID=IC;Password=inspetorchefe;";
 IDbConnection condb;
 using (condb = new SqlConnection(connectionString))
 {
    condb.Open();
    using (IDbCommand cmddb = condb.CreateCommand())
    {
        //validação de credenciais
        string sql = "SELECT * FROM [LI4-Agentes].[dbo].[Utilizador] WHERE IdUtilizador = " + idAgente;
        Console.WriteLine("\n\n" + sql);
        cmddb.CommandText = sql;
        using (IDataReader reader = cmddb.ExecuteReader())
        {
            string palavraChave = null;
            while (reader.Read())
            {
                palavraChave = (string)reader["Password"];
            }

            if (!palavraChave.Equals(pass))
            {
                var callDialog = new AlertDialog.Builder(this);
                callDialog.SetMessage("Credenciais inválidas");
                callDialog.SetNegativeButton("Ok", delegate {
                });

                callDialog.Show();
                return;
            }
        }
        reader.Close();
        // marcar como concluídas
        sql = "UPDATE [LI4-Agentes].[dbo].[Tarefa] SET Realizada=1 WHERE IdTarefa IN (" + sb.ToString() + ")";
        cmddb.CommandText = sql;
        IDataReader reader3 = cmddb.ExecuteReader();
        reader3.Close();
        //enviar os dados das tarefas
        sql = "INSERT INTO [LI4-Agentes].[dbo].[Dados] VALUES " + sbdados.ToString();
        Console.WriteLine(sbdados.ToString());
        cmddb.CommandText = sql;

        IDataReader reader2 = cmddb.ExecuteReader();
        reader2.Close();
        condb.Close();
    }
}

```

Figura 64 Excerto de código que permite estabelecer ligação para exportar para o BackOffice

6.2.7. Tarefa

Tal como expresso no requisito número 3, uma tarefa é composta por um local de investigação, objetivo, descrição, título e o caso a que é referente. Como tal, aquando da seleção de uma tarefa por parte do Agente, deve ser possível que este tenha acesso a estas informações que constituem a caracterização de uma tarefa.

Desta forma, no lado direito do ecrã principal de cada tarefa são apresentadas ao utilizador as informações mencionadas.

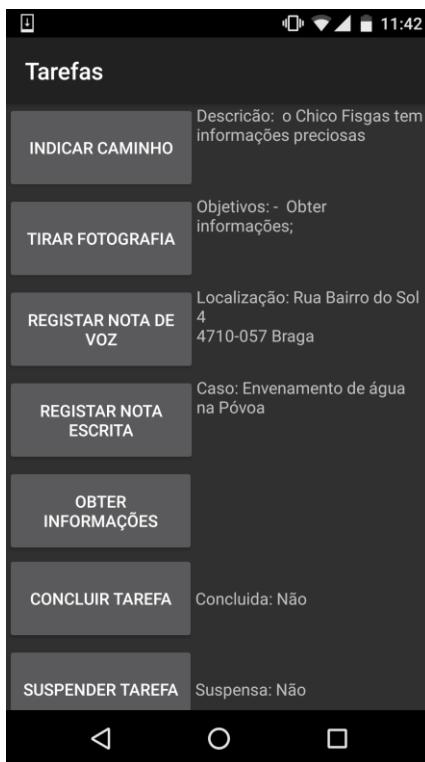


Figura 65 Ecrã principal de uma tarefa

Cada um dos campos consiste numa *TextView* que é preenchida de acordo com a tarefa que foi escolhida anteriormente no menu de seleção de tarefas.

A título de exemplo, veja-se como são completados os campos com as informações do caso, descrição, objetivos e localização de uma tarefa. O texto de cada *TextView* é alterado para o valor do campo correspondente no objeto *tarefa* que foi escolhido pelo Agente.

```
caso.Text = "Caso: " + tarefa.NomeCaso;
descricao.Text = "Descrição: " + tarefa.Descricao;
objetivos.Text = "Objetivos: " + tarefa.getObjetivos();
localizacao.Text = "Localização: " + tarefa.LocalTexto;
```

Figura 66 Exemplo de preenchimento das informações caracterizadoras de uma tarefa

No lado esquerdo do ecrã principal da tarefa são apresentados vários botões que correspondem às várias funcionalidades expressas nos requisitos.

No seguimento deste relatório, são explicadas as diversas funcionalidades que se encontram associadas à realização de uma tarefa.

6.2.8. Indicar caminho

Conforme indicado no requisito 16, deve ser possível fornecer uma orientação ao Agente sobre o local de investigação da sua tarefa, indicando-lhe o caminho a partir do sítio onde este se encontra.

O botão “Indicar Caminho” do lado esquerdo do ecrã da tarefa permite ao utilizador ter acesso a esta funcionalidade. Após a funcionalidade ser despoletada pelo *click* no botão, são obtidas as coordenadas GPS de latitude e longitude que correspondem à localização atual do Agente. Solicita-se ao utilizador que aguarde até que este processo de obtenção da sua localização esteja concluído e, no final, é-lhe possibilitado que continue para a apresentação do mapa, cujo foco de visualização já terá sido atualizado para o local de destino.



Figura 67 Obtenção da localização
atual

Neste mapa, vai ser possível ao utilizador verificar o ponto para onde se tem de deslocar (local de investigação) e o ponto onde se encontra, ligados por uma linha. É também possibilitada a interação com o mapa, sendo que o utilizador pode aproximar, desaproximar e deslocar a visualização do mapa para outros locais a partir dos respetivos gestos de toque no ecrã.

São também fornecidos diferentes modos de visualização do mapa, sendo estes os modos Terreno, Normal, Híbrido e Satélite que podem ser acedidos a partir dos botões apresentados na parte inferior do ecrã.

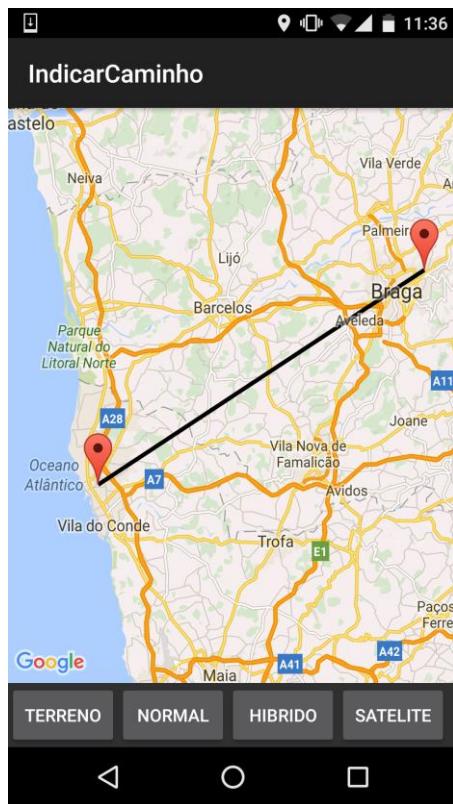


Figura 68 Indicar localizações

6.2.9. Tirar Fotografia

Conforme mencionado no requisito 17, deve ser possível que um Agente possa tirar fotografias no âmbito da realização de uma tarefa.

Desta forma, ao carregar no botão “Tirar Fotografia” no ecrã principal da tarefa, é disponibilizado um ecrã ao utilizador com a opção de abrir a câmara fotográfica do seu dispositivo móvel. Esta ação despoleta a abertura da aplicação nativa do dispositivo da câmara fotográfica e após a captura da imagem pretendida, é apresentado ao utilizador o ecrã em que se encontrava anteriormente com a apresentação da imagem registada.

Este processo envolve também a obtenção das coordenadas GPS em que o Agente se encontra de maneira a associar o registo desde dado à localização atual e o registo da data e hora no momento da captura da fotografia, conforme explicitado no requisito em causa.

No fim deste processo, a localização e a data são associados à fotografia em causa e esta é associada à respetiva tarefa.



Figura 69 Funcionalidade de
Tirar Fotografia

6.2.10. Registar Nota de Voz

Conforme indicado no requisito número 17, o Agente deve conseguir registar uma nota de voz aquando da realização de uma tarefa.

De maneira a fornecer esta funcionalidade, o botão “Registar Nota de Voz” permite a gravação de uma nota de voz.

No ecrã desta funcionalidade, são apresentados ao utilizador três botões, sendo estes o “Começar Gravação”, “Parar Gravação” e “Reproduzir”. No início, os dois últimos botões encontram-se desativados e o Agente deve pressionar o primeiro quando pretender iniciar o registo de uma gravação de voz.

```
string path = "/sdcard/Li4Detetives/gravacoes/test" + DateTime.Now.ToString("yyyyMMddHHmmssfff") + ".wav";
_start.Click += delegate {
    _stop.Enabled = !_stop.Enabled;
    _start.Enabled = !_start.Enabled;
    _rep.Enabled = false;
    _recorder.Set AudioSource(AudioSource.Mic);
    _recorder.Set OutputFormat(OutputFormat.ThreeGpp);
    _recorder.Set AudioEncoder(Encoder.AmrNb);
    _recorder.Set Outputfile(path);
    _recorder.Prepare();
    _recorder.Start();
};
```

Figura 70 Implementação do registo de uma nota de voz

Como se pode ver na imagem acima, a ação que corresponde ao *click* no botão de início de gravação de voz leva a que seja indicado o microfone do dispositivo móvel como fonte de captura de áudio e que esses dados sejam escritos no ficheiro indicado previamente.

Durante o processo de gravação, o botão de paragem deixa de estar desativado, podendo o Agente pressioná-lo quando desejar finalizar o registo da nota de voz.

```
_stop.Click += delegate {
    _stop.Enabled = !_stop.Enabled;

    _recorder.Stop();
    _recorder.Reset();

    _start.Enabled = true;
    _rep.Enabled = true;

    tarefa.Dados.Add(new Dado() { Tipo=4, Caminho = path, Latitude = latitude, Longitude = longitude,
        Data = DateTime.Now
    });
};

};
```

Figura 71 Processo de paragem da gravação de voz

Como se pode ver na imagem acima, no final do registo da nota de voz esta é associada à respetiva tarefa a partir da inserção na lista de dados.

Após este processo de registo, o botão “Reproduzir” é ativado e permite ao utilizador ouvir o registo de voz que acabou de efetuar.

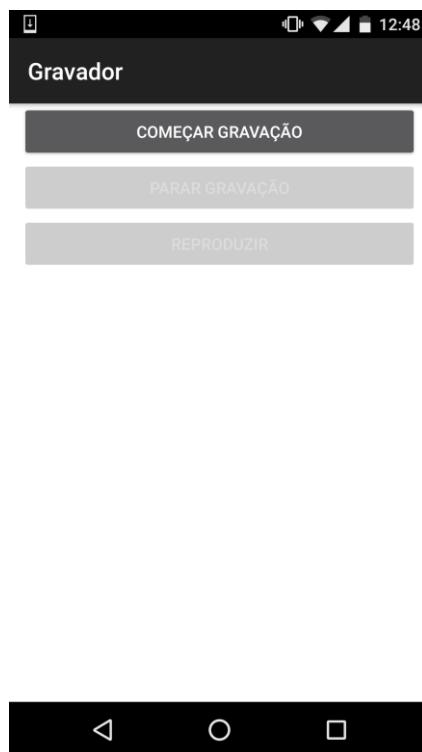


Figura 72 Registo de nota de voz

6.2.11. Registar Nota Escrita

O requisito 17 indica que um Agente deve poder registar uma nota escrita aquando da execução de uma tarefa.

O botão “Registar Nota Escrita” presente no ecrã principal da tarefa permite esta funcionalidade, na medida em que permite ao utilizador registar textualmente algo que pretenda, associando a esse registo a localização onde foi realizado e o momento em que foi escrito, ligando este dado à tarefa escolhida.

Quando o Agente escolhe esta opção, é-lhe apresentado um ecrã em que pode inserir a nota escrita que pretende sendo que, no final, esta é gravada num ficheiro, são registadas a localização e a data e é feita a ligação entre a nota escrita e a tarefa selecionada.



Figura 73 Registo de uma nota
escrita

6.2.12. Obter informações

Conforme é indicado no requisito número 18, o Agente deve ter a possibilidade de obter informações complementares sobre um determinado assunto.

Como tal, e de maneira a possibilitar este cenário, quando a opção em questão é selecionada no ecrã da tarefa, o utilizador é direcionado imediatamente para o motor de busca do Google de maneira a poder pesquisar sobre o assunto pretendido e a obter resultados para a sua pesquisa caso tenha acesso à *Internet*.

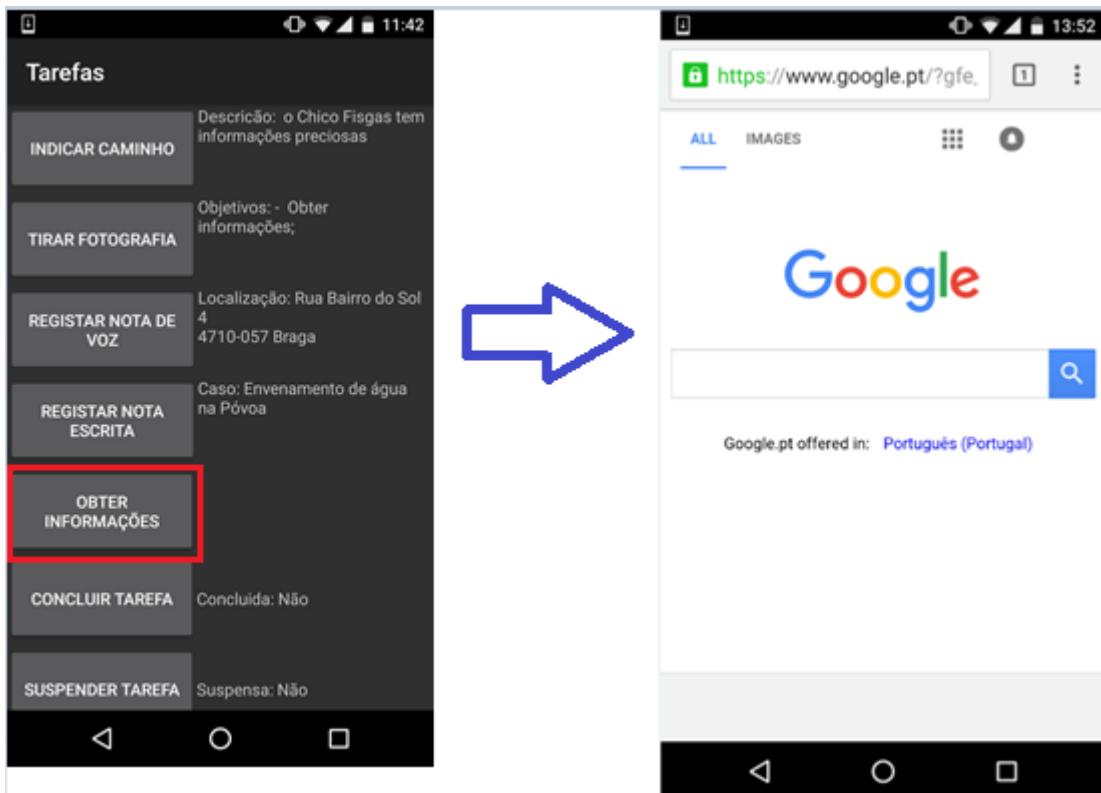


Figura 74 Funcionalidade Obter informações

6.2.13. Concluir Tarefa

O Agente deve poder indicar que uma determinada tarefa se encontra concluída a partir da alteração do seu estado. Este é um passo importante pois é um dos critérios que é verificado aquando da sincronização com o *BackOffice*. No processo de exportação das tarefas e dos seus dados, é feita uma verificação das tarefas que estejam marcadas como concluídas no *FrontOffice* (aplicação móvel) e apenas estas é que são selecionadas para exportação. Como tal, a característica de marcação de conclusão da tarefa é essencial para que a sincronização entre os dois subsistemas seja realizada de forma correta.

Do lado direito do ecrã da tarefa, pode ser visto o estado de conclusão atual da tarefa.

Abaixo encontra-se um exemplo de uma tarefa que não está marcada como concluída. Se se iniciasse um processo de exportação neste momento, a tarefa em causa não seria alvo de exportação pois internamente não está marcada como terminada.

Se o Agente manifestar intenção de marcar a tarefa como concluída, é-lhe apresentada uma caixa de diálogo em que lhe é pedida uma confirmação.



Figura 75 Tarefa a aguardar conclusão

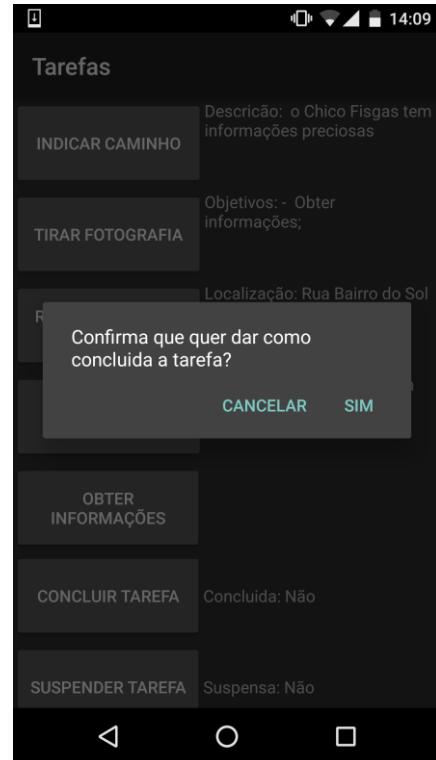


Figura 76 Caixa de diálogo - Conclusão de tarefa

6.2.14. Suspender Tarefa

De forma semelhante ao descrito anteriormente, um Agente pode também marcar uma tarefa como suspensa. O processo é bastante semelhante, sendo que se o utilizador pressionar o botão designado para o efeito, a tarefa é marcada como suspensa. Se, no entanto, o Agente pretender retomar a execução da tarefa, este pode voltar a alterar o estado desta e indicar que não continua suspensa.

Abaixo encontra-se uma imagem que ilustra as opções disponíveis para o ato de suspensão de uma tarefa.

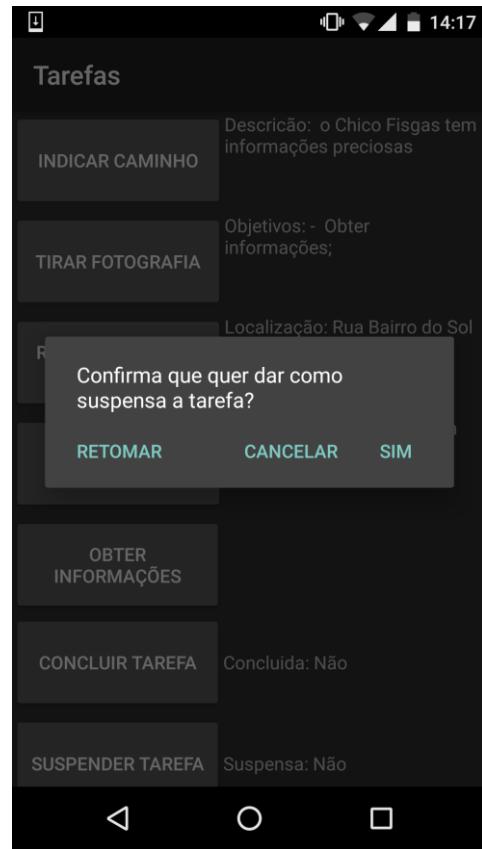


Figura 77 Opções para suspensão de uma tarefa

Quando é apresentada ao utilizador a caixa de diálogo que se pode ver na imagem acima, se este pressionar “Sim” a tarefa é marcada como suspensa, se pressionar em “Cancelar” não é efetuada nenhuma mudança no estado da tarefa e este permanece como estava e se pressionar em “Retomar” a tarefa é indicada como não estando mais suspensa.

7. Balanço do planeamento e gestão do projeto

Na fase inicial foi estabelecido um plano de trabalho que visava guiar os momentos de realização das tarefas inerentes ao projeto. Para além da decisão dos momentos em que decorreriam cada uma das tarefas, estas foram também divididas por elementos do grupo.

Nesta fase final do projeto podemos fazer um balanço global sobre o planeamento efetuado anteriormente e a forma como decorreu, na realidade, a realização das tarefas indicadas no diagrama. Desta forma, a equipa de trabalho conclui que as marcas temporais, quer de início quer de fim, da maior parte das tarefas não foram cumpridas. Isto deveu-se ao facto de a equipa de trabalho ter de gerir a execução de outros projetos em que esteve envolvida e ter de alterar os momentos em que trabalhou para o projeto em questão.

A fase que mais sofreu alterações no planeamento foi a fase de implementação, uma vez que esta começou depois do previsto.

Contudo, e apesar do planeamento inicial não ter sido cumprido na íntegra, a ordem de execução da maior parte das tarefas foi cumprida e no final, o projeto foi terminado a tempo do prazo de entrega.

8. Conclusões e trabalho futuro

Terminada esta etapa, damos por concluído a fundamentação deste projeto. Inicialmente demos especial ênfase à recolha de informações juntamente com o cliente de modo a compreender, da melhor forma possível, as suas necessidades. Adquirida esta base de conhecimento tornaram-se claras as motivações e objetivos do sistema a implementar. Assim sendo, concluímos que um investimento de trabalho nesta parte é recompensado no futuro, uma vez que uma compreensão detalhada do que constitui o problema em causa permite uma idealização e implementação mais eficazes.

A construção da maquete foi bastante útil na medida em que auxiliou o grupo a ter uma melhor visão da arquitetura e do comportamento funcional do sistema como um todo.

Relativamente ao planeamento das atividades, este permitiu ao grupo ter uma melhor percepção das tarefas que deverão ser realizadas no futuro. Para além disso, a previsão de períodos de tempo para as várias tarefas permitiu ao grupo percecionar a importância que este delineamento constitui na organização global do projeto.

No que diz respeito à segunda fase de desenvolvimento do projeto, que consiste na especificação do sistema, foram levantados os vários tipos de requisitos e elaborada a modelação do sistema através de diferentes tipos de diagramas em UML.

No que toca ao levantamento de requisitos, procedeu-se à devida separação dos mesmos em requisitos de utilizador e de sistema, sendo estes últimos divididos ainda em requisitos funcionais e não funcionais. Esta organização permitiu traduzir de melhor forma o que foi pedido pelo cliente e facilitar o entendimento das funcionalidades que o sistema deve fornecer, bem como a maneira como as deve fornecer.

Na elaboração dos modelos em UML, foram desenvolvidos vários tipos de diagramas, sendo que cada um dos tipos de diagramas escolhidos refletem aspectos diferentes sobre o projeto a realizar. O modelo de domínio, por exemplo, permitiu perceber quais os elementos que vão constituir o sistema. Os modelos de Use Cases permitem visualizar as funcionalidades a implementar e quais os utilizadores que irão usufruir das mesmas. Já os diagramas de sequência permitem refinar a descrição efetuada em cada Use Case, permitindo ter uma percepção dos vários elementos que intervêm em cada funcionalidade, bem como o ordenamento temporal das mensagens trocadas por estes elementos. A elaboração dos diagramas de atividade permitiu adicionar ainda mais informação ao que foi transmitido através dos diagramas de sequência, sendo demonstrado o fluxo de dados e especificada cada ação envolvida em cada funcionalidade. Por fim o diagrama de Classes permite obter uma perspetiva sobre a arquitetura do sistema a desenvolver, já refletindo o paradigma de programação que vai ser utilizado.

Com os modelos de sistema construídos procedeu-se à idealização e realização da base de dados em que foi feita uma análise aos requisitos levantados previamente, construindo-se assim o modelo conceitual. Com o modelo conceitual construído e validado pelo utilizador procedeu-se à passagem para o modelo lógico. Findada esta passagem, foi verificado se o

modelo estava normalizado segunda as primeiras três formas normais, foi definido um tamanho inicial da base dados e por fim, a revisão do modelo com os utilizadores.

Por fim, foram construídos *mockups* que procuram idealizar a *interface* e respetivas camadas de apresentação dos dois sistemas.

Na última fase do projeto foram implementadas as várias funcionalidades do sistema. Para além disso, foi também construída a base de dados que serve de suporte à operacionalidade quer do sistema de *BackOffice* quer do sistema de *FrontOffice*. Foi também nesta fase que se procederam aos testes necessários para verificar a operacionalidade do sistema, detetando-se eventuais erros e corrigindo-os para garantir o correto funcionamento. Por fim, procedeu-se à elaboração da documentação, que se encontra no seu estado final e relata todas as partes do projeto desde a fase de fundamentação, passando pela fase de modelação e terminando na fase de implementação.

O *software* encontra-se, então, num estado pronto a ser entregue ao cliente, assim como a documentação final.

Referências

- *eInvestigator*, 'Private Investigator App Review: Using Zillow to Prepare for Home Surveillance',
<https://www.einvestigator.com/private-investigator-app-review-using-zillow-to-prepare-for-home-surveillance/>
Zillow, Página da aplicação, <http://www.zillow.com/mobile/>
- Sommerville, I. (2000). *Software engineering*. Harlow, England: Addison-Wesley.

Lista de Siglas e Acrónimos

GPS Global Positioning System

XML eXtensible Markup Language

UML Unified Modeling Language

V.P.D Valor por defeito

Anexos

I. Anexo 1 – Exemplos de especificações de Use Case

Super Use Case		
Author	Grupo 3	
Date	25/Abr/2016 17:37:47	
Brief Description	Criação de um perfil de Agente/ Inspetor Chefe	
Preconditions	Perfil a ser criado ainda não existe;	
Post-conditions	Perfil criado com sucesso	
Flow of Events	Actor Input	System Response
	1 Solicita criação de um perfil	
	2	Pergunta que tipo de perfil pretende criar(Inspetor Chefe ou Agente)
	3 Escolhe o perfil de Agente	
	4	Solicita informações de agente
	5 Insere nome e apelido do agente	
	6 Submete dados	
	7	Processa dados inseridos
	8	Cria perfil de agente
	9	Informa sucesso
	10	Pergunta se quer associar a uma equipa
	11 Responde afirmativamente	
	12	<<Include>> Associar Agente a uma equipa
13	Informa sucesso	
Alternativa 1 [Inspetor-Chefe] (Passo 3)	Actor Input	System Response
	1 Escolhe perfil de Inspetor Chefe	
	2	Procura os Agentes que não são Inspetores-Chefe
	3	Apresenta os Agentes procurados
	4 Seleciona o Agente	
	5	Cria perfil de Inspetor Chefe
6	Volta a 13	
Excepção 2 [Responde negativamente] (Passo 11)	Actor Input	System Response
	1 Responde negativamente	
	2	Informa que o Agente não foi associado a nenhuma equipa
3		

Super Use Case	Criação de um caso	
Author	Grupo 3	
Date	25/Abr/2016 17:23:23	
Brief Description	Permite a criação no programa de um caso aceite pelo Diretor	
Preconditions	Caso ainda não existe;	
Post-conditions	Caso fica corretamente criado	
Flow of Events	Actor Input	System Response
	1 Escolhe opção para criar um caso	
	2	Solicita informações gerais sobre o caso(nome, descrição, objetivos)
	3 Insere as informações solicitadas	
	4	Verifica que as informações são válidas
	5	Informa que as informações são válidas
	6	Cria caso com as informações recolhidas
	7	Armazena caso e respetivas informações
	8	Informa sucesso
Alternativa 1 [Informações invalidas] (Passo 4)	Actor Input	System Response
	1	Verifica que as informações são inválidas
	2	Informa que as informações são inválidas
	3	Volta ao passo 2

Super Use Case	Visualizar plano de atividades	
Author	Grupo3	
Date	25/Abr/2016 17:10:44	
Brief Description	Agente deve poder consultar o seu plano de atividades a fim de o poder executar	
Preconditions	Autenticado. Existem atividades no plano	
Post-conditions	Plano de atividade visualizado	
Flow of Events	Actor Input	System Response
	1 Manifesta intenção de visualizar plano de atividades	
	2	Procura atividades
	3	Fornece atividades

II. Anexo 2 – Exemplos de diagramas de sequência

