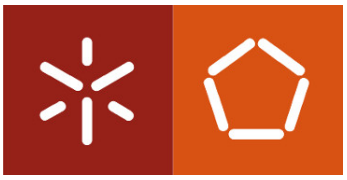

Universidade do Minho

Mestrado em Engenharia Informática

Perfil de Sistemas Inteligentes

Agentes e Sistemas Multiagentes

Rui Moreira (PG50736)
Bernardo Saraiva (PG50259)
José Gonçalves (PG50519)

18 de Março de 2023

Sistema de Gestão de Aeroporto

Conteúdo

Capítulo 1

Introdução

O presente documento surge no âmbito da Avaliação de Grupo da Unidade Curricular de Agentes e Sistemas Multiagente, constituinte do perfil de Sistemas Inteligentes.

O projeto tem por base a conceção e o desenvolvimento de um sistema multiagentes para a gestão de partidas e chegadas em aeroportos, utilizando a biblioteca *SPADE* para o desenvolvimento de agentes, juntamente com a linguagem de programação *Python*. Com isto, o objetivo passa por desenvolver agentes capazes de sensorizar e interpretar a informação percepcionada de modo a tomar decisões inteligentes de forma cooperativa.

O grupo começou por realizar uma análise à estrutura sugerida pela equipa docente, descrita no capítulo 2, e apresentou todas as discordâncias com a mesma. Após isto, descreveu-se a arquitetura do sistema multiagente pretendido, tirando partido de toda a modelação *UML*, apresentada no capítulo 3.

Depois desta primeira fase, passou-se para a implementação em código, onde no capítulo 4 apresenta-se a estrutura da solução, e no capítulo 5 apresenta-se a explicação da implementação para cada agente com as suas variáveis e *behaviours*. Efetuou-se também uma análise aos resultados obtidos, descrita no capítulo 6.

Por fim, este relatório termina com uma conclusão, capítulo 7, onde é feita uma análise crítica ao trabalho desenvolvido e onde são apresentadas algumas sugestões e recomendações para a melhoria do sistema produzido.

Capítulo 2

Análise da estrutura sugerida

A abordagem inicial passou por uma análise crítica à estrutura sugerida no enunciado, tendo sido efetuadas algumas alterações que o grupo considerou que faria mais sentido.

Deste modo, em vez de implementar 4 agentes, decidiu-se implementar 5 agentes, acrescentando apenas um agente que permite introduzir aviões, periodicamente, no sistema.

No enunciado foi feita uma sugestão relativamente ao Gestor de Gares, a qual sofreu algumas alterações, nomeadamente com o envio da informação sobre as gares à torre de controlo no início do seu processo, justificando-se pelo facto de a torre de controlo não precisar da informação sobre as gares para o seu funcionamento.

Relativamente à sugestão acerca da pretensão de os aviões descolarem, em que é dito que a torre de controlo deve indicar ao gestor de gares qual a pista que o avião pode utilizar, considerou-se mais coeso ser a torre de controlo a comunicar a pista para o avião descolar, diretamente com o próprio.

Capítulo 3

Modelação UML

Com o desenvolvimento do presente projeto, foi desenvolvida uma modelação UML, que não só permitiu maior clareza e compreensão do projeto, como identificar antecipadamente problemas e riscos. Em seguida apresentam-se os vários diagramas desenvolvidos, juntamente com uma explicação dos mesmos.

3.1 Diagrama de Classes

Para responder aos objetivos propostos, o grupo pretendeu construir um sistema multiagente composto por 5 tipos de agentes, Avião, Criador de Aviões, Torre de Controlo, Gestor de Gares e Agente de Informação.

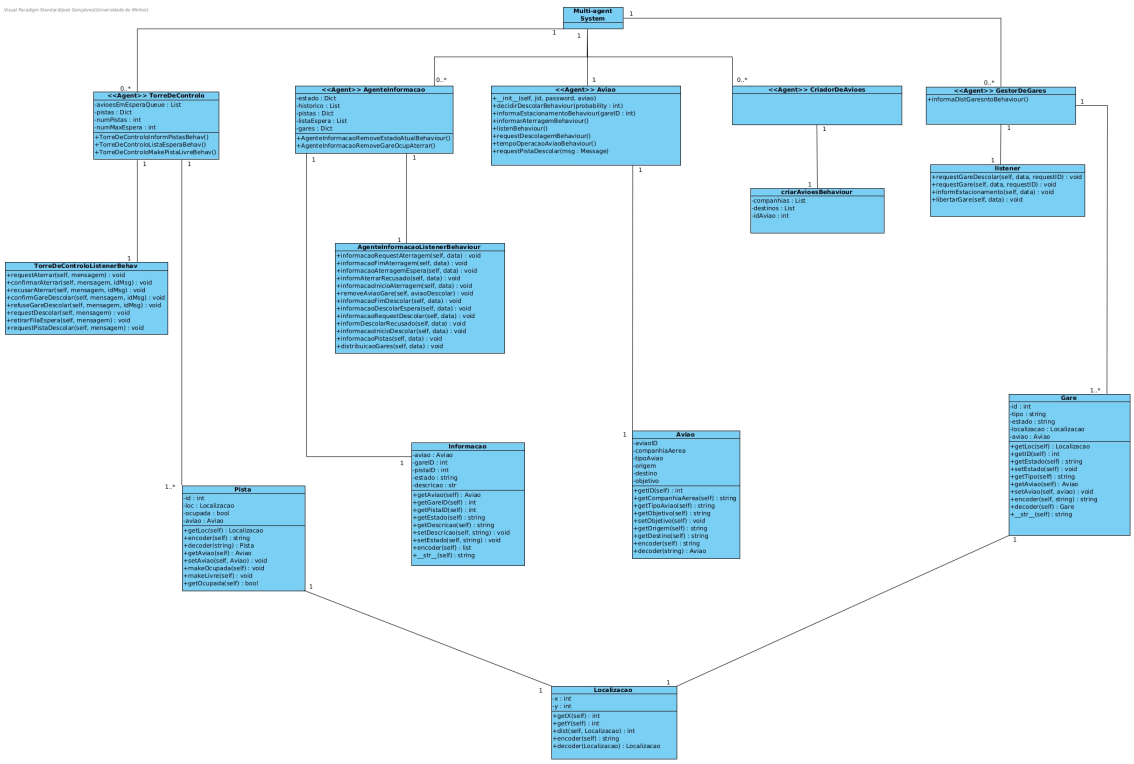

Figura 3.1: Diagrama de Classes

O avião, como agente principal de um aeroporto pretende aterrar e descolar. Como podemos

ver pela figura acima, este contém informações como id do avião, companhia, avião de transporte de mercadorias ou comercial, origem, destino e objetivo, que estão guardadas numa classe *Aviao*. Para conseguir ter aviões constantemente a chegar ao aeroporto, o grupo decidiu criar um agente que utiliza um PeriodicBehaviour para criar um agente Avião periodicamente.

A Torre de Controlo representa o agente responsável pela gestão dos aviões que pretendem aterrar ou descolar no aeroporto. Para isto, contém toda a informação das pistas, das listas de espera tanto para aterrar como descolar, assim como o número máximo de aviões que podem estar em espera para aterrar. De notar que não existe um número máximo que limite a fila de espera para descolar.

O Gestor de Gares, como o próprio nome revela, é o agente responsável pela gestão das gares e por tratar solicitações de aviões que pretendam estacionar numa gare. Este agente, quando iniciado, cria um número de gares e guarda-as num dicionário.

Por fim o Agente de Informação tem o objetivo de apresentar toda a informação geral sobre o estado do aeroporto. Para isto, este agente está permanentemente a receber informações da Torre de Controlo, através de um *CyclicBehaviour*.

Todos os *Behaviours* de cada tipo de agente serão explicados em detalhe no capítulo 5.

3.2 Diagrama de Comunicação

Primeiramente é necessário entender, de um modo mais abstrato, o fluxo de trocas de mensagens entre os agentes Torre de Controlo, Avião e Gestor de Gares. Para isso, recorreu-se a dois diagramas de comunicação, um para o processo de aterrar e outro para o de descolar de um avião.

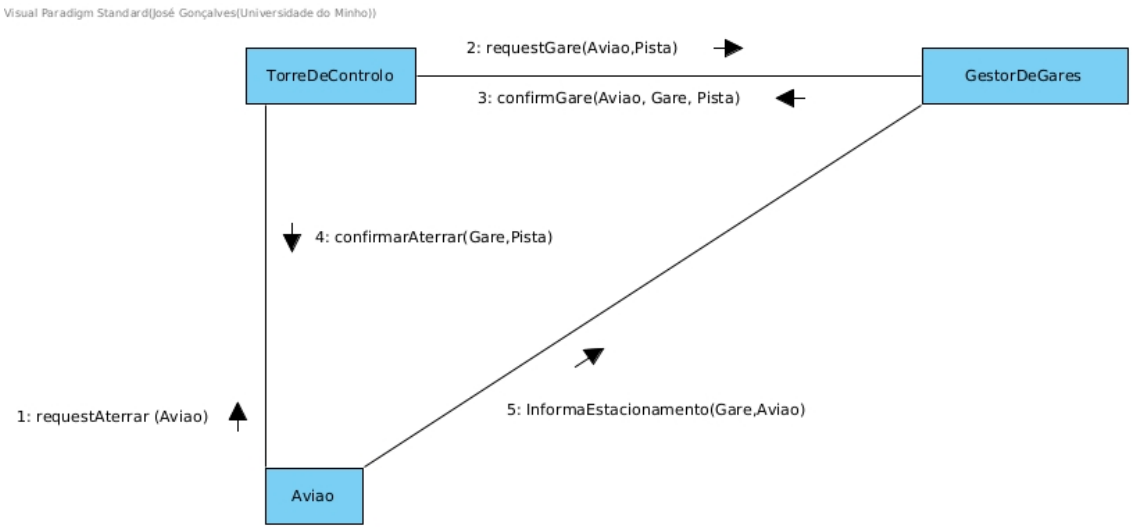

Figura 3.2: Diagrama de Comunicação Aterrar

Como é possível ver pela Figura 3.2, o agente Avião começa por enviar uma mensagem à Torre de Controlo a pedir para aterrar. A Torre de Controlo atribui uma pista e manda um pedido de gare ao Gestor de Gares com a informação do avião e da pista atribuída, para este lhe atribuir uma gare adequada ao tipo de avião (comercial/mercadorias) e a mais próxima à pista atribuída. De seguida, a Torre de Controlo confirma a aterragem ao Avião, indicando-lhe a gare e a pista atribuídas. Por fim, o avião quando aterra, informa o estacionamento ao Gestor de Gares.

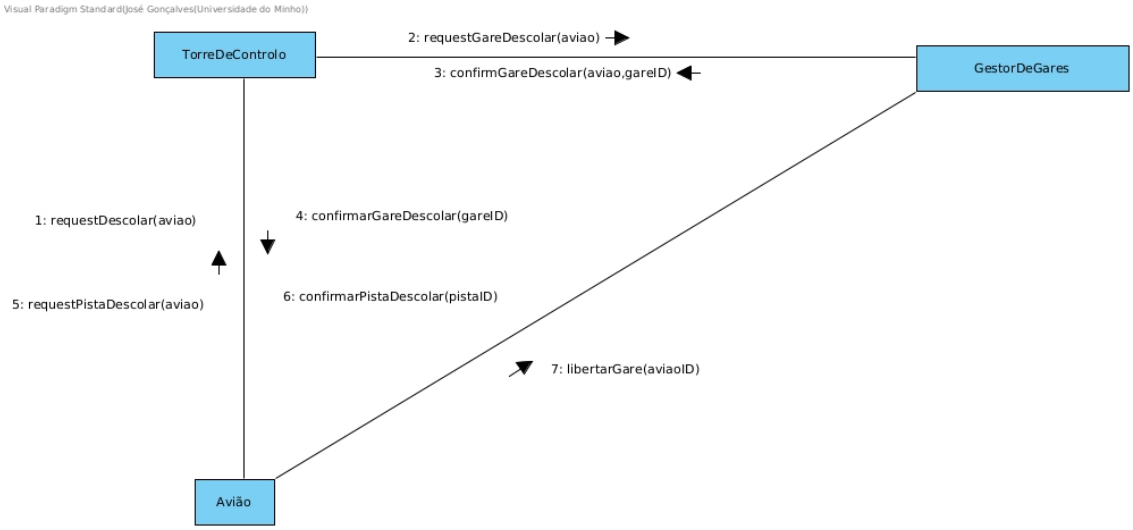

Figura 3.3: Diagrama de Comunicação Descolar

Já o processo de comunicação de descolar, representado na Figura 3.3, começa com o Avião a pedir uma gare à Torre de Controlo e esta reencaminha a mensagem ao Gestor das Gares. Neste caso, visto que os aviões antes de descolar estão num parque de estacionamento, este responde à Torre de Controlo com a gare mais próxima à localização do estacionamento, e esta volta a reencaminhar a informação para o avião. Após o tempo de operação na gare, o Avião pede à Torre de Controlo uma pista para descolar, ao que este lhe responde afirmativamente com o id da pista. Após receber esta informação, o avião envia mensagem ao Gestor das Gares para informar que vai libertar a sua gare e descolar.

3.3 Diagrama de Atividades

Relativamente ao esquema de atividades, estas foram divididas naquelas que são as duas principais tarefas a executar: Aterragem e Descolagem.

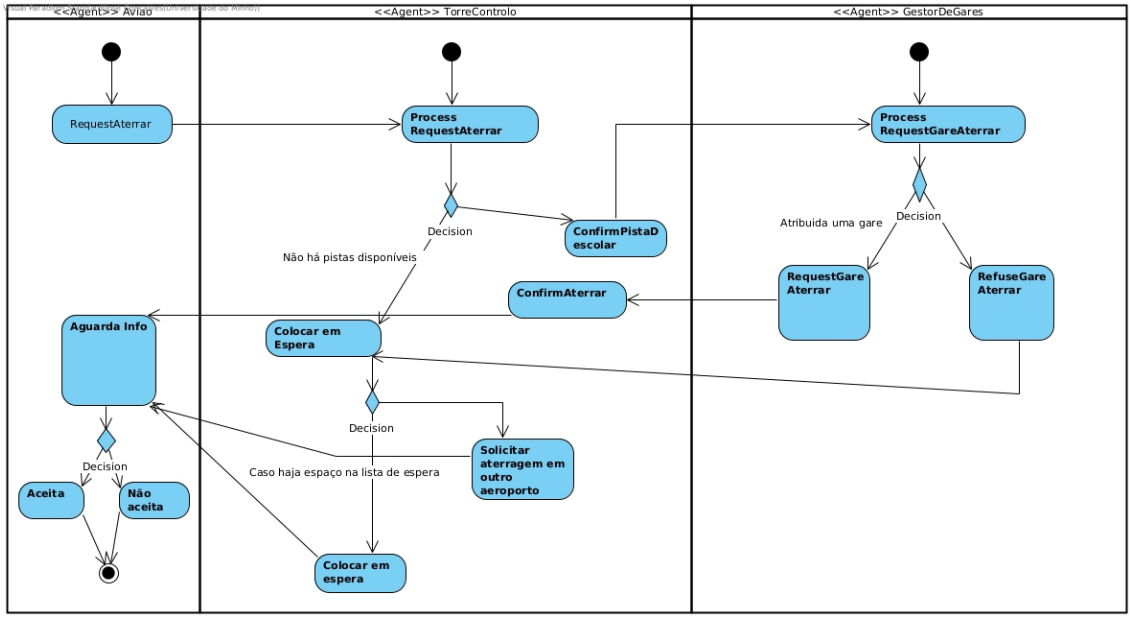

Figura 3.4: Diagrama de atividades Aterrar

Neste sentido, o diagrama de atividades relativo à aterragem, Figura 3.4, ilustra o seu início, sendo dado por parte de um Agente Avião, que envia uma mensagem a solicitar a aterragem à Torre de Controlo. Com isto, a Torre de Controlo verifica se há disponibilidade de pistas para a aterragem do avião e, em caso afirmativo, pede ao Gestor de Gares para verificar a disponibilidade de gare. Em caso de não haver pistas, esta solicita ao avião que aguarde novas informações, podendo esperar para que aterre naquele mesmo aeroporto, ou, em caso da lista de espera estar cheia, em outro aeroporto.

Já por parte do Gestor de Gares, que é contactado pela Torre a pedir uma gare, verifica se pode atribuir uma gare que esteja disponível para o tipo do avião que requisitou, atribuindo a gare mais próxima da pista. Caso não seja possível, informa a Torre que o avião terá que aguardar.

Já o avião, após aguardar a resposta ao seu pedido de aterragem, efetua a ação que lhe foi concedida permissão.

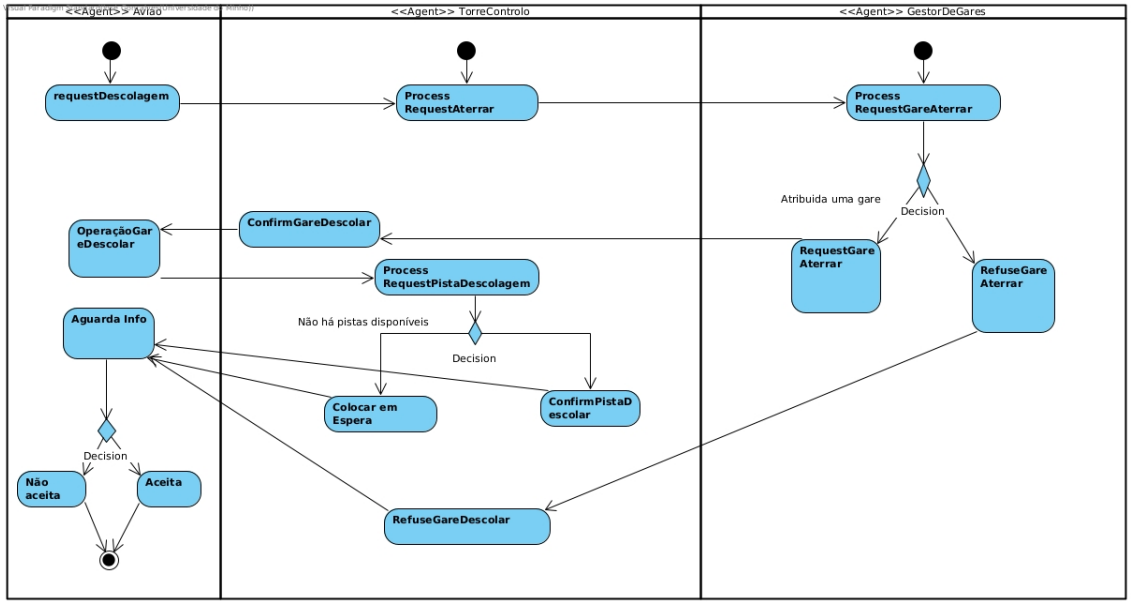

Figura 3.5: Diagrama de atividades Descolar

De forma inversa ao que foi referido acima, o avião poderá também solicitar permissão para descolar. Então, como é possível verificar na Figura 3.5, envia uma mensagem à Torre de Controlo e esta reencaminha para o Gestor de Gares, que verifica se pode ou não atribuir uma gare, de modo a que o avião vá para lá durante o seu tempo de operação (a gare atribuída é sempre a mais próxima do estacionamento onde ele se encontra). Caso haja disponibilidade, o Gestor de Gares confirma à Torre a informação e esta retransmite ao avião. Em seguida, o avião, após realizar a operação na gare, envia um pedido de pista para descolar à Torre de Controlo, que, por sua vez, trata de verificar se pode reservar uma pista para o avião. Caso não haja disponibilidade de pistas ou de gares, o avião é informado, sendo acrescentado à lista de espera.

Finalmente, a decisão é comunicada ao avião, gerando assim a sua nova ação, com base nas condicionantes referidas.

3.4 Diagrama de Sequência

Por fim, o grupo construi um diagrama de Sequência, ilustrado abaixo na Figura 3.6, que permite ver com mais detalhe, todo o fluxo de mensagens desencadeadas por todas as decisões enquanto um avião está no sistema desde, listas de espera cheias, pistas ocupadas, gares ocupadas, etc., ou seja, desde o pedido de aterrar até descolar.

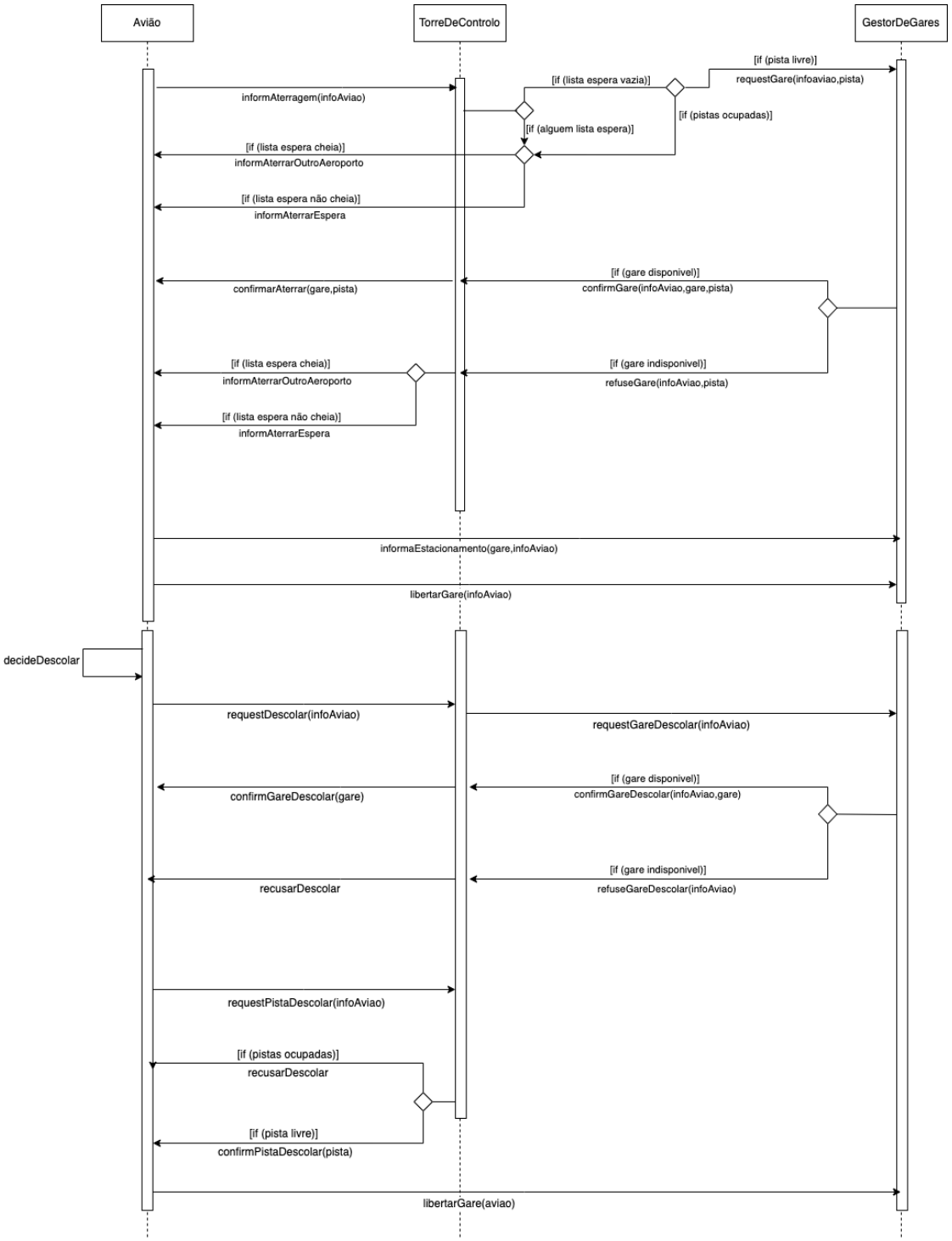

Figura 3.6: Diagrama de Sequência

Capítulo 4

Estrutura da Solução

No desenvolvimento do projeto a organização consistiu em 3 partes:

- Agents: Conjunto dos ficheiros dos Agentes que operam no sistema
- *Behaviours*: Conjunto dos *Behaviours*, que está dividido numa pasta para cada um dos Agentes
- Utilitários: Conjunto de ficheiros utilitários que constituem classes, e em cada um, contêm funções que permitem dar *encode* ou *decode* dos mesmos

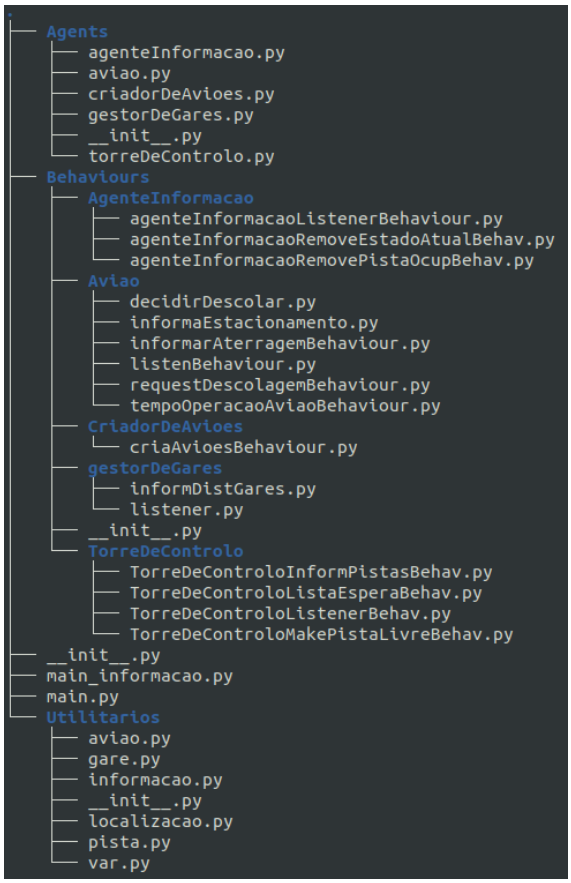

Figura 4.1: Estrutura do projeto

Capítulo 5

Implementação dos Agentes

Para implementar tudo o que foi modelado acima, nesta secção serão apresentados todos os elementos constituintes dos Agentes, entre outros, variáveis e *behaviours*. Ao longo do desenvolvimento, cada mensagem enviada entre agentes é constituída por um código como prefixo, seguido de uma barra vertical, que permite indentificar a mensagem trocada entre os agentes. Relativamente às *performatives* usadas, estas foram utilizadas dos tipos *request*, *confirm*, *refuse* e *inform*, pelo que os seus nomes são bastante explicativos e o prefixo da mensagem indica o tipo utilizado, juntamente com a sua funcionalidade, que aparece logo em seguida.

5.1 Avião

O agente Avião foi o primeiro agente a ser desenvolvido, sendo este o principal elemento do sistema. Deste modo, começou-se por desenvolver uma classe Avião, constituinte do conjunto de ficheiros Utilitários, que contém os dados de cada avião, funcionando como um construtor. Esta classe é constituída por:

- aviaoID - identificador do avião;
- companhiaAerea - companhia aérea do avião;
- tipoAviao - tipo de avião, podendo ser comercial ou de mercadorias;
- origem - aeroporto de origem do avião;
- destino - aeroporto de destino do avião;
- objetivo - objetivo do avião, podendo ser aterrar ou descolar

Para além dos *getters* e *setters* para cada um dos atributos, a classe utilitária de Avião contém também um método de *encode* e *decode*, que facilitam o envio das mensagens entre os agentes. Deste modo, o método *encode* converte um objeto Avião numa String, assim como o método *decode* converte uma String numa instância da classe Avião.

Relativamente ao agente Avião, este implementa um conjunto de *Behaviours*, que são responsáveis por receber as mensagens enviadas pelo agente Torre de Controlo, e por enviar as mensagens tanto para o agente Torre de Controlo, de modo a que este possa organizar o tráfego aéreo, como para o Gestor de Gares para informar estacionamento. De notar que a comunicação do Agente Avião é maioritariamente com o Agente Torre de Controlo.

Behaviours

- **informarAterragemBehaviour:** Ao iniciar o Agente Avião, é chamado o *Behaviour informarAterragemBehaviou*r, sendo este um *OneShotBehaviour*, que tem como objetivo enviar uma mensagem para o agente Torre de Controlo, para solicitar autorização para aterragem.

- **listenBehaviour:** É também inicializado o *Behaviour listenBehaviour*, que permite manter o Agente à escuta de mensagens provenientes de outros agentes (neste caso sendo a comunicação restrita à Torre de Controlo). Este *Behaviour* é de caráter Cíclico, sendo que o Agente Avião está sempre à espera de mensagens. Este listenBehaviour é responsável por informações de confirmação/negação de aterragens e movimento entre gares.

- **informaEstacionamento:** Este *Behaviour* de caráter *OneShot* é responsável por informar o agente Gestor de Gares que o avião aterrou, e que está pronto para estacionar. Com este *Behaviour*, é enviada uma mensagem para o agente Gestor de Gares, com o objetivo de informar que irá ocupar a gare, sendo também iniciado o *Behaviour* tempoOperaçãoBehaviour, que faz com que passado o tempo de operação liberte a gare.

- **tempoOperacaoBehaviour:** Para esta funcionalidade foi implementado um *TimeoutBehaviour*, ao qual é fornecido um valor de timeout que ocorre um determinado tempo depois do momento em que é chamado *(datetime.now() + timedelta(seconds=TEMPO_OPERACAO_GARE))*. Assim, tem como objetivo libertar a gare e chamar o *Behaviour* decidirDescolarBehaviour.

- **decidirDescolarBehaviour:** O *Behaviour* decidirDescolarBehaviour (com caráter Periódico) tem o objetivo de controlar a descolagem do avião, sendo que este tem uma probabilidade associada, que determina se o avião descola ou não. Caso a probabilidade verifique um determinado valor, é permitido que o avião descole, sendo iniciado um novo *Behaviour* com o nome requestDescolagemBehaviour, de caráter *OneShot*. Caso contrário, o avião fica em espera durante um determinado tempo, e volta a tentar descolar.

- **requestDescolagemBehaviour:** Este *Behaviour* de caráter *OneShot* é responsável por pedir ao agente Torre de Controlo uma gare para operacionar e descolar.

- **requestPistaDescolagemBehaviour:** Este *Behaviour* de caráter *TimeoutShot* é disparado quando o avião finalizou a operação na gare para descolar e assim, tem como objetivo enviar um pedido ao agente Torre de Controlo a pedir uma pista para descolar.

5.2 Criador de Aviões

O agente Criador de Aviões é responsável por criar os aviões ao longo do tempo, sendo que estes são criados com informações aleatórias, com um determinado intervalo de tempo entre a criação de cada avião, sendo este definido na variável *INTERVALO_CHEGADA_AVIOES* presente no ficheiro *var.py*.

Para a criação dos aviões, foi implementado um *Behaviour* de caráter Periódico, criaAviõesBehaviour, que tem como objetivo criar um novo avião, com todos os seus atributos escolhidos aleatoriamente.

5.3 Gestor de Gares

O agente Gestor de Gares é responsável por gerir as gares de estacionamento, sendo que este tem como objetivo receber pedidos de estacionamento de aviões e atribuir uma gare a cada avião, de modo a que este possa estacionar, ou responder a pedidos de libertar gare. De modo contrário, caso não existam gares disponíveis, o avião fica em espera até que uma gare fique disponível (no caso de aterrar, dependendo se a lista de espera estiver cheia ou não).

Este agente conta com uma variável que indica o número de gares e pistas disponíveis. Para além disso, conta também com um dicionário que contém todas as gares, sendo que cada gare no dicionário, de modo análogo ao Avião, é uma classe Gare (ficheiro na pasta Utilitários), que contém as seguintes informações:

- id - identificador da gare;

- tipo - tipo de gare, podendo ser destinada a aviões comerciais ou de mercadorias;

- estado - estado da gare, podendo ser livre ou ocupada;
- localizacao - localização da gare, a qual é representada por um objeto da classe Localizacao, sendo este mais um dos Utilitários implementados;
- aviao - avião que está estacionado na gare (representado por um objeto do tipo Avião), sendo que caso a gare esteja livre, o valor deste atributo é None.

Para além dos *getters* e *setters* para cada um dos atributos, a classe utilitária da Gare contém também um método de *encode* e *decode*, que facilitam o envio das mensagens entre os agentes.

Adicionalmente, para que o agente Gestor de Gares possa gerir as gares, este conta com um conjunto de Behaviours, que são responsáveis por tratar as mensagens trocadas e, deste modo, gerir as gares.

Behaviours

- **informDistGares:** Este *behaviour* de caráter *OneShot* é responsável por informar o Agente de Informação acerca de todas as informações das gares, sendo que este é iniciado quando o agente Gestor de Gares é iniciado.
- **listener:** Este *behaviour* de caráter Cíclico é responsável por receber as mensagens relativas à gestão das gares, sendo que estas podem ser de pedir uma gare, libertar uma gare ou informar estacionamento de um avião.

5.4 Torre De Controlo

O agente Torre de Controlo é responsável por gerir o tráfego aéreo, sendo que este tem como objetivo receber pedidos de descolagem e aterragem de aviões, e atribuir uma pista a cada avião, de modo a que este possa descolar ou aterrar.

Este agente conta com uma variável que indica o número de pistas disponíveis e uma variável que indica o número máximo de aviões em espera para aterrar, assim como a lista de espera de aviões para descolar e aterrar.

Para além disso, de modo análogo aos anteriores, conta também com um dicionário que contém as pistas, sendo que cada pista corresponde a uma classe, que contém as seguintes informações:

- id - identificador da pista;
- loc - A localização da pista, sendo que esta é representada por um objeto da classe Localização, que por sua vez é mais um dos Utilitários implementados;
- ocupada - indicador do estado da pista, podendo esta estar livre ou ocupada;
- aviao - avião que está a descolar ou a aterrar na pista (representado por um objeto do tipo Avião), sendo que caso a pista esteja livre, o valor deste atributo é None.

Para que possa ser efetuada a gestão do aeroporto por parte da Torre de Controlo, esta conta com um conjunto de Behaviours, que são responsáveis por tratar as mensagens trocadas e, deste modo, gerir as pistas:

Behaviours

- **TorreDeControloListenerBehav:** Este *behaviour* de caráter Cíclico é responsável por receber as mensagens relativas à gestão das pistas, sendo que estas podem ser de pedir uma pista, libertar uma pista ou informar que uma pista está livre. Este *behaviour* também efetua a ponte entre a comunicação relativa às gares até aos aviões, e vice-versa.

- **TorreDeControloListaDeEsperaBehav:** Através do seu caráter Cíclico, este *behaviour* é responsável por gerir a lista de espera de aviões para aterrar, sendo que este é iniciado quando o agente Torre de Controlo é iniciado. Periodicamente, este verifica se tem condições para algum avião aterrrar ou descolar e, em caso afirmativo, chama um avião da lista de espera para aterrar ou descolar. De notar que, se a lista de espera de descolar for maior que a de aterrar, ele dá prioridade à lista de espera descolar. Caso contrário, dá prioridade à lista de espera aterrar.
- **TorreDeControloMakePistaLivreBehav:** Quando a Torre de Controlo ocupa uma pista, é chamado este *behaviour* de caráter *TimeoutBehaviour* (ocorrendo após o TEMPO_OPERACAO_PISTA definido), que tem como objetivo libertar a pista e notificar de que está livre. Assim, este avisa o agente de informação de que o avião já aterrou ou descolou naquela determinada pista.
- **TorreDeControloInformPistasBehav:** Este *OneShotBehaviour* é responsável por informar o Agente de Informação acerca de toda a informação sobre as pistas, sendo que este é iniciado quando o agente Torre de Controlo é iniciado.

5.5 Agente de Informação

O agente de informação é responsável por receber e enviar informação relativa ao estado do aeroporto, sendo que este tem como objetivo receber pedidos de informação de aviões, e enviar a informação relativa às gares e pistas disponíveis.

Este agente conta com as seguintes variáveis:

- estado: dicionário que contém o estado de cada avião, ou seja se for atualizada uma informação de uma avião que já lá se encontra, esta será atualizada;
- histórico: dicionário que contém o histórico de cada avião, ou seja se for atualizada uma informação de uma avião que já lá se encontra, esta será adicionada ao histórico, não sendo nunca removida ou atualizada;
- pistas: dicionário que contem a informação relativa às pistas, sendo que esta é atualizada sempre que uma pista é libertada ou ocupada;
- listaEsperaAterrar: lista que contém os aviões que estão em espera para aterrar;
- listaEsperaDescolar: lista que contém os aviões que estão em espera para descolar;
- gares: dicionário que contém a informação relativa às gares, sendo que esta é atualizada sempre que uma gare é libertada ou ocupada.

A sua classe utilitária (Informação) contém toda a informação, servindo como um objeto de troca de mensagens entre os agentes, sendo que esta contém a informação relativa aos aviões, gares e pistas, contendo o **aviao**, o **ID da gare**, o **ID da pista**, o **estado** e a **descrição**.

Behaviours

- **AgenteInformacaoListenerBehaviour:** Este *behaviour* de caráter Cíclico é responsável por receber as mensagens relativas à informação dos aviões, sendo que estas podem ser de pedir informação, atualizar informação ou informar que um avião aterrou ou descolou.
- **AgenteInformacaoRemoveEstadoAtualBehav:** Com este *behaviour*, após um *timeout* de *TEMPO_ESTADO*, é removido o estado atual de um avião que a sua informação não seja atualizada à mais de *TEMPO_ESTADO*;
- **AgenteInformacaoRemoveGareOcupAterrarBehav:** De igual modo ao anterior, este *behaviour* permite aplicar um *Timeout* para que liberte a gare após o tempo de operação da Gare, definido pela variável *TEMPO_OPERACAO_GARE*;

Capítulo 6

Resultados obtidos e análise crítica

Com a finalização do sistema, foram efetuados vários testes com diferentes valores, de modo a avaliar a correção e robustez do sistema. Foi também aplicado um sistema de cores, em que cada agente possui uma cor representativa, de modo a facilitar a leitura e percepção das mensagens.

O teste apresentado em seguida foi realizado com as seguintes variáveis globais:

```
TEMPO_OPERACAO_PISTA = 10
TEMPO_OPERACAO_GARE = 10
TEMPO_ESTADO = 10
NUM_PISTAS = 2
NUM_MAX_ESPERA = 5
NUM_GARES = 10
INTERVALO_CHEGADA_AVIOES = 5
ESTACIONAMENTO = Localizacao(NUM_GARES*0.7 + 1,(NUM_PISTAS + 2)/2)
PROBABILITY = 30 # prob de decidir descolar
```

Figura 6.1: Variáveis globais

, resultando na seguinte organização do aeroporto:

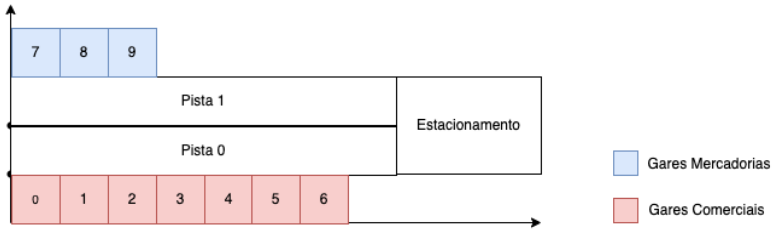

Figura 6.2: Organização do Aeroporto

, do qual se obteve os seguintes resultados:

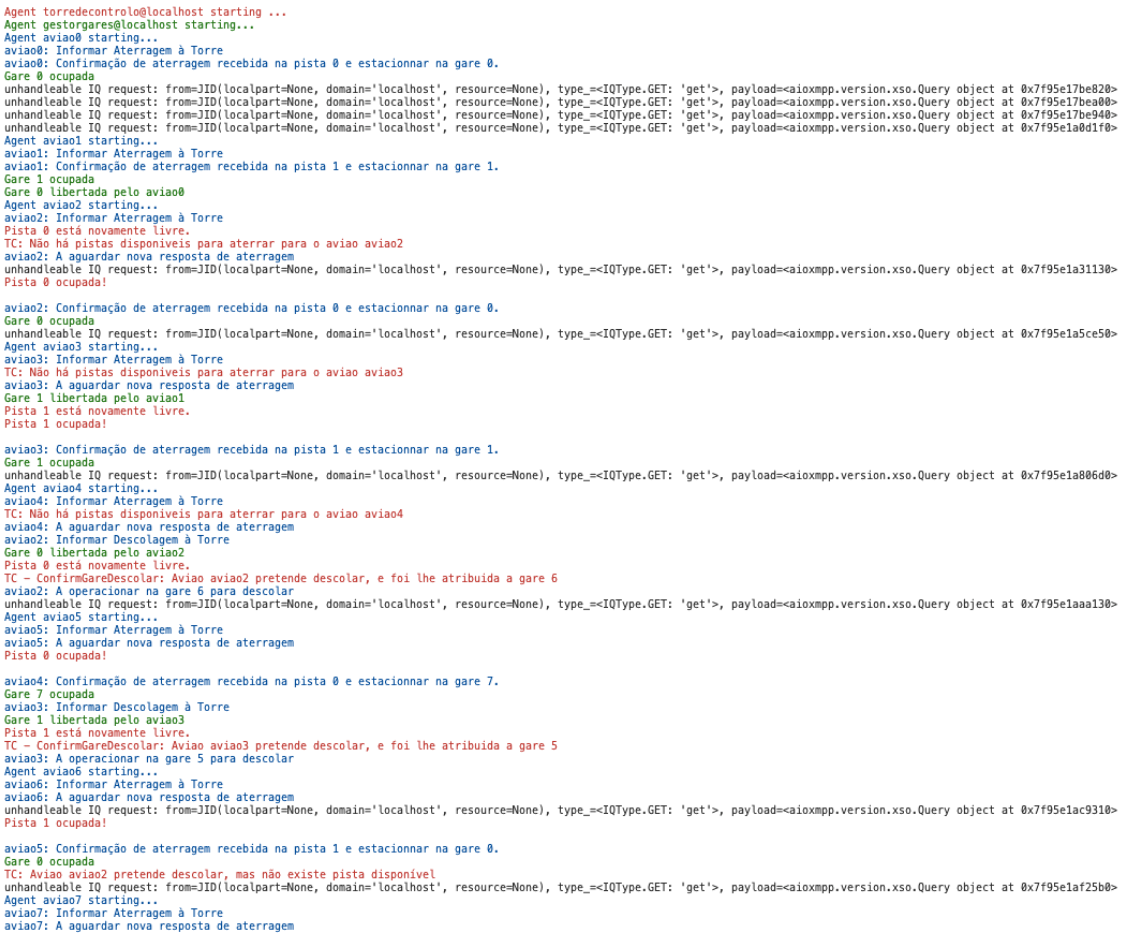

Figura 6.3: Logs gerados ao longo da troca de mensagens entre os agentes

6.1 Informação representada pelo agente de informação

Implementou-se vários menus no agente de informação, para tornar mais fácil a navegação entre as várias opções de informação. Todos estes resultados abaixo foram obtidos no fim do teste acima.

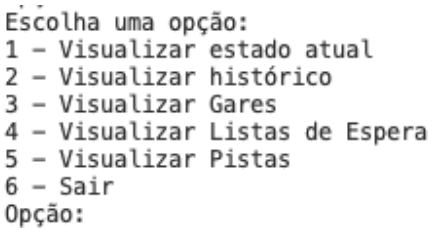

Figura 6.4: Menu Principal

Começando pelo menu principal, tem-se 5 opções:

1. Redireciona para o menu do Estado (Figura 6.8)
2. Redireciona para o menu do Histórico (Figura 6.12)
3. Exibe o estado das Gares e por qual avião está ocupado

Gare	AviaoID	Companhia	Tipo	Origem	Destino
0	-	-	-	-	-
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	-
4	-	-	-	-	-
5	aviao3	TAP	Comercial	Atlanta	Lisboa
6	aviao2	EASYJET	Comercial	Al Garhoud	Pequim
7	-	-	-	-	-
8	-	-	-	-	-
9	-	-	-	-	-

Figura 6.5: Menu Estado Gares

4. Exibe o estado da lista de espera de aterrar, bem como de descolar

Aterrar

Ordem de Chegada	Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião
0	aviao6	Dammam	Barcelos	SATA	Mercadorias
1	aviao7	Dammam	Amarante	TAP	Comercial

Descolar

Ordem de Chegada	Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião
0	aviao2	Al Garhoud	Pequim	EASYJET	Comercial

Figura 6.6: Menu Etado Lista de Espera

5. Exibe o estado das pistas e por qual avião está ocupado

Pista ID	Localização	Ocupada	Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião
0	0,1	True	aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias
1	0,2	True	aviao5	Barcelos	Lisboa	SATA	Comercial

Figura 6.7: Menu Estado Pistas

```
Escolha uma opção:
1 - Visualizar estado geral
2 - Visualizar estado das chegadas
3 - Visualizar estado das partidas
0 - Voltar Menu Principal
Opção:
```

Figura 6.8: Menu Estado

Passando para o menu Estado, que se encontra dentro do menu Principal, este tem 3 opções:

1. Exibe o estado atual de todos os aviões no sistema

Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião	Gare ID	Pista ID	Estado	Descrição
aviao0	Paris	Munique	RYANAIR	Comercial	0	0	Feito	Aterrar
aviao1	Dammam	Cancún	FLY EMIRATES	Comercial	1	1	Feito	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	–	FilaDeEsperaPista	Descolar
aviao3	Atlanta	Lisboa	TAP	Comercial	5	–	Operacao	Descolar
aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias	7	0	A acontecer	Aterrar
aviao5	Barcelos	Lisboa	SATA	Comercial	0	1	A acontecer	Aterrar
aviao6	Dammam	Barcelos	SATA	Mercadorias	–	–	FilaDeEspera	Aterrar
aviao7	Dammam	Amarante	TAP	Comercial	–	–	FilaDeEspera	Aterrar

Figura 6.9: Menu Estado Geral

2. Exibe o estado atual de chegadas

Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião	Gare ID	Pista ID	Estado	Descrição
aviao0	Paris	Munique	RYANAIR	Comercial	0	0	Feito	Aterrar
aviao1	Dammam	Cancún	FLY EMIRATES	Comercial	1	1	Feito	Aterrar
aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias	7	0	A acontecer	Aterrar
aviao5	Barcelos	Lisboa	SATA	Comercial	0	1	A acontecer	Aterrar
aviao6	Dammam	Barcelos	SATA	Mercadorias	–	–	FilaDeEspera	Aterrar
aviao7	Dammam	Amarante	TAP	Comercial	–	–	FilaDeEspera	Aterrar

Figura 6.10: Menu Estado Chegadas

3. Exibe o estado atual de partidas

Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião	Gare ID	Pista ID	Estado	Descrição
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	–	FilaDeEsperaPista	Descolar
aviao3	Atlanta	Lisboa	TAP	Comercial	5	–	Operacao	Descolar

Figura 6.11: Menu Estado Partidas

```
Escolha uma opção:
1 - Visualizar histórico geral
2 - Visualizar histórico de um aviao
0 - Voltar Menu Principal
```

Figura 6.12: Menu Histórico

Por fim, tem-se o menu Histórico, que contém 2 opções:

1. Mostra o histórico geral de todos os aviões

Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião	Gare ID	Pista ID	Estado	Descrição
aviao0	Paris	Munique	RYANAIR	Comercial	–	–	Pediu	Aterrar
aviao0	Paris	Munique	RYANAIR	Comercial	0	0	A acontecer	Aterrar
aviao1	Dammam	Cancún	FLY EMIRATES	Comercial	–	–	Pediu	Aterrar
aviao1	Dammam	Cancún	FLY EMIRATES	Comercial	1	1	A acontecer	Aterrar
aviao0	Paris	Munique	RYANAIR	Comercial	0	0	Feito	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	–	–	Pediu	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	–	–	FilaDeEspera	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	0	0	A acontecer	Aterrar
aviao3	Atlanta	Lisboa	TAP	Comercial	–	–	Pediu	Aterrar
aviao3	Atlanta	Lisboa	TAP	Comercial	–	–	FilaDeEspera	Aterrar
aviao1	Dammam	Cancún	FLY EMIRATES	Comercial	1	1	Feito	Aterrar
aviao3	Atlanta	Lisboa	TAP	Comercial	1	1	A acontecer	Aterrar
aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias	–	–	Pediu	Aterrar
aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias	–	–	FilaDeEspera	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	0	0	Feito	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	–	–	Pediu	Descolar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	–	Operacao	Descolar
aviao5	Barcelos	Lisboa	SATA	Comercial	–	–	FilaDeEspera	Aterrar
aviao4	Amesterdão	Al Garhoud	SATA	Mercadorias	7	0	A acontecer	Aterrar
aviao3	Atlanta	Lisboa	TAP	Comercial	1	1	Feito	Aterrar
aviao3	Atlanta	Lisboa	TAP	Comercial	–	–	Pediu	Descolar
aviao3	Atlanta	Lisboa	TAP	Comercial	5	–	Operacao	Descolar
aviao6	Dammam	Barcelos	SATA	Mercadorias	–	–	FilaDeEspera	Aterrar
aviao5	Barcelos	Lisboa	SATA	Comercial	0	1	A acontecer	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	–	FilaDeEsperaPista	Descolar
aviao7	Dammam	Amarante	TAP	Comercial	–	–	FilaDeEspera	Aterrar

Figura 6.13: Menu Histórico Geral

2. Mostra o histórico de um determinado avião

Avião ID	Origem	Destino	Companhia Aérea	Tipo de Avião	Gare ID	Pista ID	Estado	Descrição
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	-	-	Pediu	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	-	-	FilaDeEspera	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	0	0	A acontecer	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	0	0	Feito	Aterrar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	-	-	Pediu	Descolar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	-	Operacao	Descolar
aviao2	Al Garhoud	Pequim	EASYJET	Comercial	6	-	FilaDeEsperaPista	Descolar

Figura 6.14: Menu Histórico de determinado Avião

Pela análise das figuras acima, é possível verificar que toda informação está congruente com o que é apresentado no output dos *logs* gerados ao longo da troca de mensagens entre os agentes (Figura 6.3).

Por exemplo, analisando as trocas de mensagens do *aviao2* na Figura 6.3 :

1. *aviao2* pediu para aterar
2. *TC* rejeitou o pedido e colocou avião2 na lista de espera
3. *TC* chama avião2 para aterrar na pista 0 e gare 0
4. *aviao2* aterra e liberta pista e gare
5. *aviao2* pede para descolar à *TC*
6. *TC* pede uma gare ao *GG* e reencaminha a mensagem com a gare 6 para o *aviao2*
7. *aviao2* faz operação na gare 6
8. *aviao2* pede pista para descolar à *TC*
9. *TC* responde que não existe pista disponível e o *aviao2* fica à espera de pista para descolar

Esta comunicação corresponde à informação armazenada no histórico do *aviao2*, Figura 6.14. No fim da comunicação entre os agentes, pode-se concluir que o *aviao2* estará na gare 6, como se verifica na Figura 6.5, e estará na lista de espera para descolar, como se verifica na Figura 6.6.

Capítulo 7

Conclusão

Com a realização do presente projeto prático, foi possível aplicar e consolidar os diversos conceitos abordados na *UC* ao longo do semestre, nomeadamente os conceitos de Agente e modelação com os seus *Behaviours*, assim como a modelação *UML* para estruturar o projeto a ser desenvolvido.

Destaca-se a modularidade presente no resultado final, sendo as funções de cada um dos agentes bastante bem isoladas, assim como o realismo obtido na simulação, no sentido em que nunca é apresentado o mesmo resultado, por conta da implementação de ações baseadas em resultados probabilísticos. A quantidade de tabelas e informações apresentadas também constitui um ponto positivo, no sentido em que o utilizador consegue ter uma visão total e detalhada sobre os acontecimentos passados e atuais do sistema.

Algo que poderia ser melhorado no sistema desenvolvido seria a implementação de uma interface gráfica mais completa, onde fosse possível, por exemplo, visualizar todas as movimentações do avião nas ações de aterrar, descolar e estacionar.