

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

**DSS Fase 3**

**Grupo Nº 4**

João Correia (A84414)

Marco Pereira (A89556)

Pedro António (A58062)

Rúben Cerqueira (A89593)

23 de dezembro de 2020



João



Marco



Pedro



Rúben

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>5</b>
<b>2</b>	<b>Alterações à Segunda Fase</b>	<b>7</b>
2.1	Diagrama de classes . . . . .	7
2.1.1	Package: Utilizador . . . . .	8
2.1.2	Package: Robo . . . . .	8
2.1.3	Package: Stock . . . . .	8
2.1.4	Adições de Packages e classes integrantes . . . . .	8
2.1.5	Classe Mapa . . . . .	9
<b>3</b>	<b>Data Access Objects</b>	<b>10</b>
3.1	Gestores . . . . .	11
3.2	Notificações . . . . .	11
3.3	Paletes . . . . .	11
3.4	Prateleiras . . . . .	12
3.5	Robôs . . . . .	12
3.6	Robôs Transportadores . . . . .	12
3.7	Rotas . . . . .	13
<b>4</b>	<b>Diagrama de Máquina de estados</b>	<b>14</b>
4.1	Estado: Login . . . . .	15
4.2	Estado: Menu Principal . . . . .	15
4.3	Estado: Mapa em Tempo Real . . . . .	16
4.4	Estado: Localização de Paletes . . . . .	16
<b>5</b>	<b>Diagrama de Instalação</b>	<b>17</b>
5.1	Sender de códigos QR . . . . .	18
5.2	Leitor de códigos QR . . . . .	18
5.3	Robo de transporte . . . . .	18
5.4	Servidor do Armazém . . . . .	18
<b>6</b>	<b>Implementação da Aplicação</b>	<b>19</b>
6.1	Comunicar código QR . . . . .	19
6.2	Sistema comunica ordem de transporte . . . . .	20
6.3	Notificar recolha de Paletes . . . . .	21
6.4	Notificar entrega de Paletes . . . . .	22
6.5	Consultar listagem de localizações . . . . .	22

<b>7</b>	<b>Análise Crítica</b>	<b>24</b>
<b>8</b>	<b>Conclusão</b>	<b>26</b>

# **Lista de Figuras**

2.1	Diagrama de Classes alterado . . . . .	7
4.1	Diagrama de Máquina de Estados . . . . .	14
5.1	Diagrama de Instalação . . . . .	17
6.1	Diagrama de sequência - Comunicar código QR . . . . .	20
6.2	Diagrama de sequência - Sistema comunica ordem de transporte . . . . .	21
6.3	Diagrama de sequência - Notificar recolha de paletes . . . . .	21
6.4	Diagrama de sequência - Notificar entrega de Paletes . . . . .	22
6.5	Diagrama de sequência - Consultar listagem de localizações . . . . .	23
6.6	Diagrama de Sequência atualizaSistema . . . . .	23

# Capítulo 1

## Introdução

O presente relatório visa apresentar a terceira e última fase do trabalho proposto no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software.

Esta fase final do trabalho visa a concretização de todo o planeamento efetuado na fase 1 e 2 do projeto.

Reiterando o processo que leva ao ponto de partida desta fase 3:

Na primeira fase do projeto elaboraram-se o **Modelo de Domínio** e os **Modelos de Use Cases**, responsáveis por dar uma vista geral sobre a forma como as diferentes entidades do sistema se vão relacionar, e qual é a funcionalidade que viria a ser implementada posteriormente.

A implementação da segunda fase foi efetuada sobre a base construída na fase anterior, e começou a descrever a aplicação a ser desenvolvida com um nível de detalhe já bastante superior. No decorrer desta fase produziram-se os seguintes diagramas: **Diagrama de Componentes**, **Diagrama de Classes**, assim como os **Diagramas de Sequência**.

Tendo em posse toda esta planificação, e uma visão muito clara do modo como se quer desenvolver a aplicação que é o objetivo do projeto global, a Terceira Fase do projecto vai focar-se na conversão do planeamento para código Java, a fim de se produzir uma aplicação que cumpra com todos os requisitos especificados.

Ao longo do presente relatório vamos detalhar todos os passos tomados para a escrita do código que compõe a nossa aplicação, qual a funcionalidade implementada, e por fim, uma visão crítica do resultado obtido.

Houve múltiplas reformulações às funcionalidades pedidas, assim como uma reestruturação do Armazém que se pretende modelar para este Trabalho Prático, e começaremos assim por indicar quais as alterações a ter em conta e quais as soluções adoptadas.

Os **Use Cases** a considerar são então os seguintes:

- Comunicar código QR (Actor: Leitor de códigos QR)
- Sistema comunica ordem de transporte (Actor: Robot / Iniciativa: Sistema)
- Notificar recolha de paletes (Actor: Robot)
- Notificar entrega de paletes (Actor: Robot)
- Consultar listagem de localizações (Actor: Gestor)

A implementação anterior visava a modelação integral do sistema de funcionamento do Armazém, desde a chegada de um pedido, até à entrega da matéria prima encomendada para esta sair do Armazém.

Analizando os **Use Cases** a considerar, esta começa agora com a comunicação do Código QR quando as paletes com matéria prima chegam ao armazém, simula o processo de armazenamento efetuado pelos Robots, e disponibiliza ao Gestor as opções de ver o funcionamento do Armazém em tempo real, assim como obter uma listagem de todas as paletes dentro do recinto.

Outras alterações que também condicionaram a implementação presente:

- Não há agora distinção entre matérias primas perecíveis e não perecíveis
- A interface é agora no formato textual
- O Mapa foi simplificado

Dadas estas mudanças, consideramos pertinente começar por detalhar as soluções encontradas pelo grupo para adaptar o projeto que se iria desenvolver com a modelação entregue na segunda fase ao que é agora proposto.

# Capítulo 2

# Alterações à Segunda Fase

O Armazém que se pretende modelar sofreu alterações, resultando numa necessidade de voltar a rever todo o planeamento desenvolvido ao final da segunda fase do projeto. Esta necessidade advém de ser preciso ter um ponto de partida que reflita a aplicação que se quer modelar, e é crucial que este novo planeamento seja feito antes de se começar com a escrita do código.

Começaremos assim por mostrar o Diagrama de Classes no qual se baseou a implementação da aplicação, e, de seguida, detalhar que mudanças foram efetuadas, bem como o seu propósito.

## 2.1 Diagrama de classes

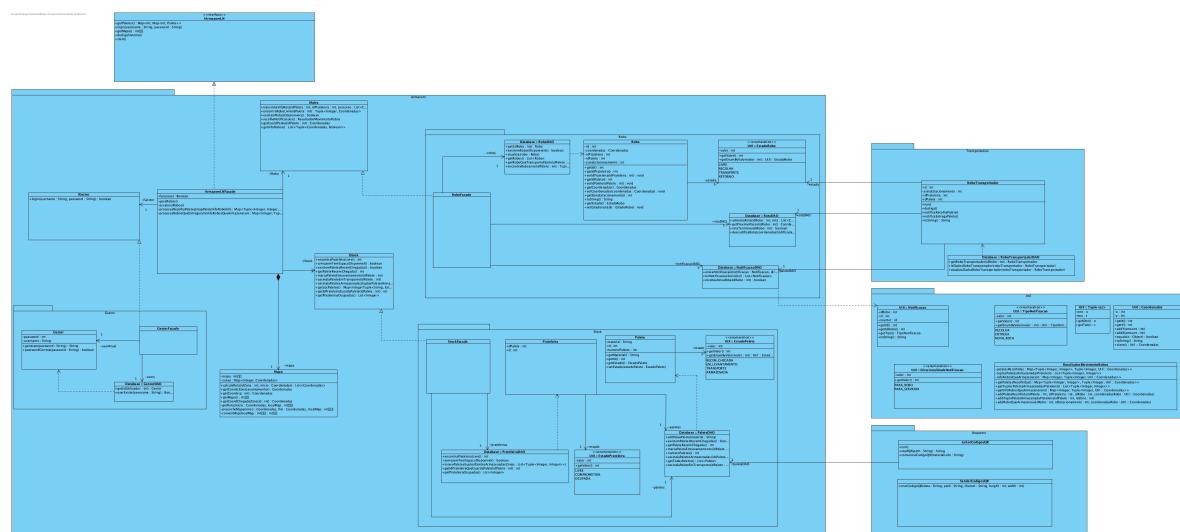


Figura 2.1: Diagrama de Classes alterado

Em relação ao **Diagrama de Classes** entregue na segunda fase, foram feitas as seguintes alterações:

### 2.1.1 Package: Utilizador

Dado que agora apenas vamos ter um tipo de utilizador, o Gestor, começámos por mudar o nome do Package, que agora se intitula "Gestor", para que haja clareza na sua função.

Dada a inexistência de uma entidade Encarregado, a funcionalidade de verificação de códigosQR foi transferida para o Subsistema Stock.

Uma vez que a fase da atividade do Armazém que estamos a modelar começa com a leitura do códigoQR já não chegarão ao Armazém Pedidos de Entrega ou Requisição, e consequentemente a sua aceitação foi também removida do Package.

### 2.1.2 Package: Robo

Verificou-se uma mudança de filosofia em relação ao funcionamento dos robôs, sendo os robôs agora entidades físicas e externas ao sistema. Como tal, os robôs presentes neste Package serão apenas representações dos verdadeiros robôs interpretáveis pelo sistema. Os verdadeiros robôs (RoboTransporte) encontrar-se-ão num package separado, denominado de Transportation.

Para garantir o acesso à Base de Dados por parte do Robo, foram criadas três classes de acesso, sendo estas o **RoboDAO**, **RotaDAO** e **NotificacaoDAO**. Estas adições serão justificadas mais pormenoradamente num capítulo posterior.

Foi removida a classe **Etapa**, devido às alterações feitas na classe **Mapa**, que serão abordadas em conjunto com as supramencionadas.

### 2.1.3 Package: Stock

Tal como no caso do Package **Robo**, foram adicionadas classes de acesso à Base de Dados relativas a informação de **Prateleiras** e **Paletes** do sistema.

Foi removida a associação entre as classes **Robo** e **Palete**, sendo esta substituída por um atributo na classe **Robo** correspondente ao id da palete que está a transportar no momento.

Por fim, foi removida a classe **CodigoQR**, uma vez que esta referência não será relevante para o sistema. A única função desta será a de permitir estabelecer a comunicação entre o Leitor de códigos QR e o sistema, para efetuar a inserção de Paletes no armazém.

### 2.1.4 Adições de Packages e classes integrantes

Para fazer a distinção de entidades externas ao sistema, e para adicionar classes auxiliares, foram criados novos packages, separados da lógica de negócio. Estes novos Packages

denominam-se **Util**, **Transportation** e **Requests**.

O Package **Util** conterá classes auxiliares para a clareza e eficiência do código.

O Package **Transportation** contém as classes que representam a implementação física do Robo, sendo esta um ator do sistema.

Finalmente, o Package **Requests** relaciona-se com a criação e leitura de códigos QR, contendo as classes SenderCodigosQR e LeitorCodigosQR, que estarão responsáveis pelo registo de novas Paletes no sistema.

### 2.1.5 Classe Mapa

Após análise posterior, concluiu-se que era mais simples a manipulação do Mapa, estando este implementado sob a forma de uma matriz (contrariamente à anterior realização desta como grafo), para mais fácil representação e análise do sistema. Com esta aproximação, tornou-se irrelevante a classe **Etapa**, mencionada acima, uma vez que esta estava relacionada com a elaboração do Mapa em grafo.

Além da representação do Mapa em forma de matriz, também foi necessário adicionar um mapeamento de id de zona para **Coordenadas**, para se ter acesso às coordenadas das diversas zonas constituintes do Mapa.

## Capítulo 3

# Data Access Objects

Uma das grandes diferenças face à segunda fase do projecto é a transposição de todas as estruturas que seriam guardadas em memória para Data Access Objects (DAOs).

DAOs são classes que nos permitem separar a lógica de negócio e a lógica de persistência uma da outra. A lógica de negócio conhece a API do DAO, mas não tem contacto com nenhuma das operações que são feitas internamente, nem no modo como os dados são depois guardados. Desde modo, o DAO funciona como uma "caixa negra", em que providencia todos os dados pedidos à lógica de negócio, e pode ser mudado livremente para outra implementação, sem por isso afetar o funcionamento da aplicação, desde que implemente a mesma API.

No nosso projecto, de modo a que fosse possível o armazenamento de informação numa Base de Dados, recorreu-se à biblioteca JDBC. Para o efeito, decidiu-se que iríamos implementar DAOs para todos os dados que fossem passíveis de ser alterados no decurso do funcionamento da aplicação, e que não queiramos ver perdidos quando esta é encerrada. Ficámos assim com a seguinte lista de DAOs:

- Gestores
- Notificações
- Paletes
- Prateleiras
- Robôs
- Robôs transportadores
- Rotas

### 3.1 Gestores

O DAO correspondente aos **Gestores** representará uma tabela composta pelas seguintes colunas:

- username
- password (criptada)

Esta classe será utilizada no momento de login.

### 3.2 Notificações

O DAO correspondente às **Notificações** representará uma tabela composta pelas seguintes colunas:

- id
- idRobo
- tipo
- direcionalidade

Esta classe será utilizada como forma de comunicação bidirecional entre sistema e robôs, sendo a coluna idRobo correspondente ou ao robô destinatário, ou ao que envia a Notificação ao sistema. A coluna tipo será representativa do tipo de notificação, e a direcionalidade indicativa se a Notificação se encontra destinada a um robô, ou ao sistema.

### 3.3 Paletes

O DAO correspondente às **Paletes** representará uma tabela composta pelas seguintes colunas:

- id
- material
- estado

Esta classe é utilizada tanto no processo de registo de entrada de paletes no sistema, despontado pelo Leitor de códigos QR, como também em toda a lógica de processamento de notificações dos Robôs, onde se vai alterando o estado da Paleta. O estado da Paleta poderá assumir um de quatro valores: Recém chegada, em levantamento, em transporte e armazenada.

### 3.4 Prateleiras

O DAO correspondente às **Prateleiras** representará uma tabela composta pelas seguintes colunas:

- id
- estado
- idPalete

Esta classe será utilizada no escalonamento de Robos, de forma encontrar uma Prateleira disponível para uma Paleta recém chegada e no processamento das Notificações de entrega de Paleta enviados pelo Robo. Poderá ter um de três estados: livre, comprometida (situação que ocorre quando se encontra um Robô a transportar uma Paleta até à Prateleira sendo que a Prateleira ainda não se encontra ocupada) e ocupada. Será composta também pelo seu id e id da Paleta que armazena.

### 3.5 Robôs

O DAO correspondente aos **Robôs** será a forma do sistema criar uma representação na camada de negócio dos Robôs, que funcionam remotamente de forma externa e agnóstica ao sistema. A tabela correspondente será composta pelas seguintes colunas:

- id
- estado
- idPalete

A classe será utilizada exclusivamente pelo sistema e terá como propósito comunicar aos robôs informações que o sistema pretende que os robôs reais alterem em si (como, por exemplo, o id da Paleta que carregam), assim como obter as coordenadas reais de cada Robô, verificar disponibilidades de transporte, etc...

### 3.6 Robôs Transportadores

Este DAO serve como forma dos Robôs reais acederem à Base de Dados. Como tal, partilharão a tabela \*Robo\* com o RoboDAO acima mencionado, que tem a seguinte estrutura:

- id
- estado
- idPalete

A classe será utilizada exclusivamente pelos Robôs reais, sendo que nunca é chamada pelo sistema, de forma a se obter total independência do mesmo. Servirá para o Robô real ler e atualizar a sua informação, como, por exemplo, as suas coordenadas atuais, ou o id da Paleta que deve carregar, atributos que são alterados na própria tabela \*Robo\*, visto que não são indicados nas notificações.

### 3.7 Rotas

Este DAO serve como forma de guardar as Rotas atribuídas a cada Robô, sob forma textual. A tabela correspondente será composta pelas seguintes colunas:

- idRobo
- valor

Esta classe é utilizada pelo sistema para indicar a cada Robô que rota deve seguir no momento do cálculo da Rota. Cada Robô irá, depois, usar esta classe para saber quais as coordenadas que deve tomar na iteração seguinte do seu movimento.

## Capítulo 4

# Diagrama de Máquina de estados

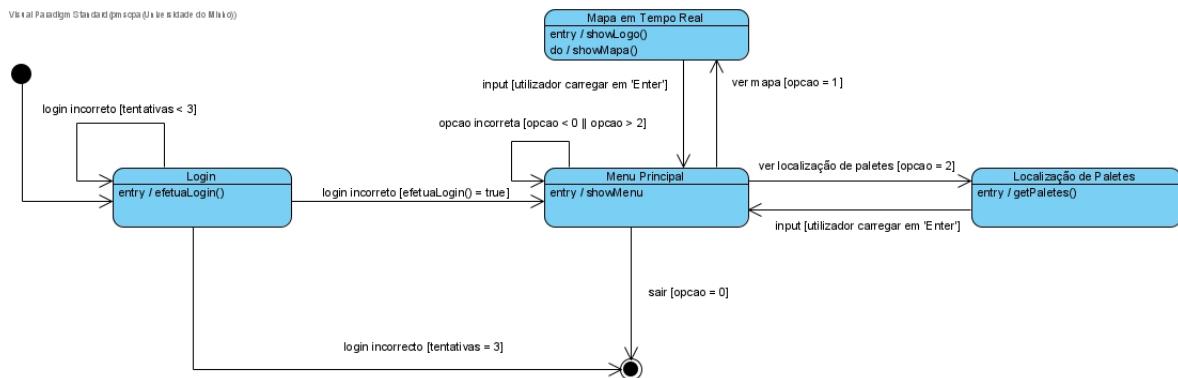


Figura 4.1: Diagrama de Máquina de Estados

Dada a existência de uma interface em que o utilizador navega de tela para tela, considerou-se pertinente fazer a sua representação através de um **Diagrama de Máquina de Estados**.

Um **Diagrama de Máquina de Estados** é um diagrama que visa detalhar o comportamento dinâmico de um elemento. Para tal, são mostrados todos os diferentes estados que um objeto pode tomar, e as transições entre eles.

Um estado é uma condição do elemento em que este desempenha uma função, ou aguarda algum evento. Transições são as ligações que existem entre estados, e descrevem qual o evento que tem que acontecer para que o elemento transite entre os dois estados.

Passaremos seguidamente à análise de cada um dos estados presentes no diagrama.

## 4.1 Estado: Login

O primeiro estado que o utilizador encontra é o de Login, em que ao entrar é chamado o método efetuaLogin(), que permite que o utilizador insira os seus dados de acesso à aplicação.

Desta inserção acontecerá um de três eventos:

- **os dados de acesso estão incorretos, e o utilizador errou menos de 3 vezes** - é pedido que introduza novamente os dados.
- **os dados de acesso estão incorretos, tendo esgotado as 3 tentativas** - para assegurar a segurança do sistema e para evitar tentativas de acesso não autorizadas, a aplicação termina caso sejam inseridas 3 combinações incorretas de username/password.
- **o par username/password está correto** - a interface passa para um segundo estado, o do Menu Principal.

## 4.2 Estado: Menu Principal

No Menu Principal, é mostrado o menu ao entrar, e o utilizador pode fazer várias opções consoante a informação que quiser consultar:

- **Opcão < 0 || Opcão > 2** - A opção não existe, e é pedido novamente input ao utilizador.
- **Opcão = 1** - A opção 1 corresponde à alteração do estado para um estado que permite a visualização do Mapa do Armazém em tempo real, e que será descrito abaixo.
- **Opcão = 2** - Esta opção altera o estado, disponibilizando ao utilizador a informação sobre todas as paletes existentes neste momento no armazém. Este estado será também tratado abaixo.
- **Opcão = 0** - Quando o utilizador seleciona a opção 0, a aplicação é encerrada.

### **4.3 Estado: Mapa em Tempo Real**

No estado de Mapa em Tempo Real, ao entrar é disponibilizado o logo do armazém, e depois, de segundo em segundo, o ecrã é atualizado com o estado atual de todo o armazém. Neste, incluem-se movimentações dos Robôs em tempo real, que são identificados de diferente modo consoante estejam a carregar Paletes ou no seu percurso de volta, assim como todas as Prateleiras, que indicam com ícones diferentes se contêm ou não Paletes.

A única opção que está disponível ao utilizador neste estado, é a de carregar na tecla "Enter" para voltar ao estado de Menu Principal. Qualquer outro input é ignorado.

### **4.4 Estado: Localização de Paletes**

Este é o último estado que pode ser tomado. É acedido através da inserção da opção 2 no Menu Principal, e a sua função é a de mostrar ao utilizador toda a informação relacionada com as Paletes presentes no armazém.

Esta é disponibilizada na forma de uma tabela, em que consta a ID de cada Paleta, o material que esta contém, assim como o seu estado. O estado da paleta pode ser:

- RECEM CHEGADA - se ainda estiver a aguardar para ser disponibilizada pra recolha pelos Robos
- EM ESPERA - caso esteja já colocada na zona de recolha e à espera que o Robo designado a recolha.
- EM TRANSPORTE - este estado significa que a Paleta está neste momento a ser transportada por um Robo
- ARMAZENADA - a Paleta já se encontra devidamente armazenada numa Prateleira

# Capítulo 5

## Diagrama de Instalação

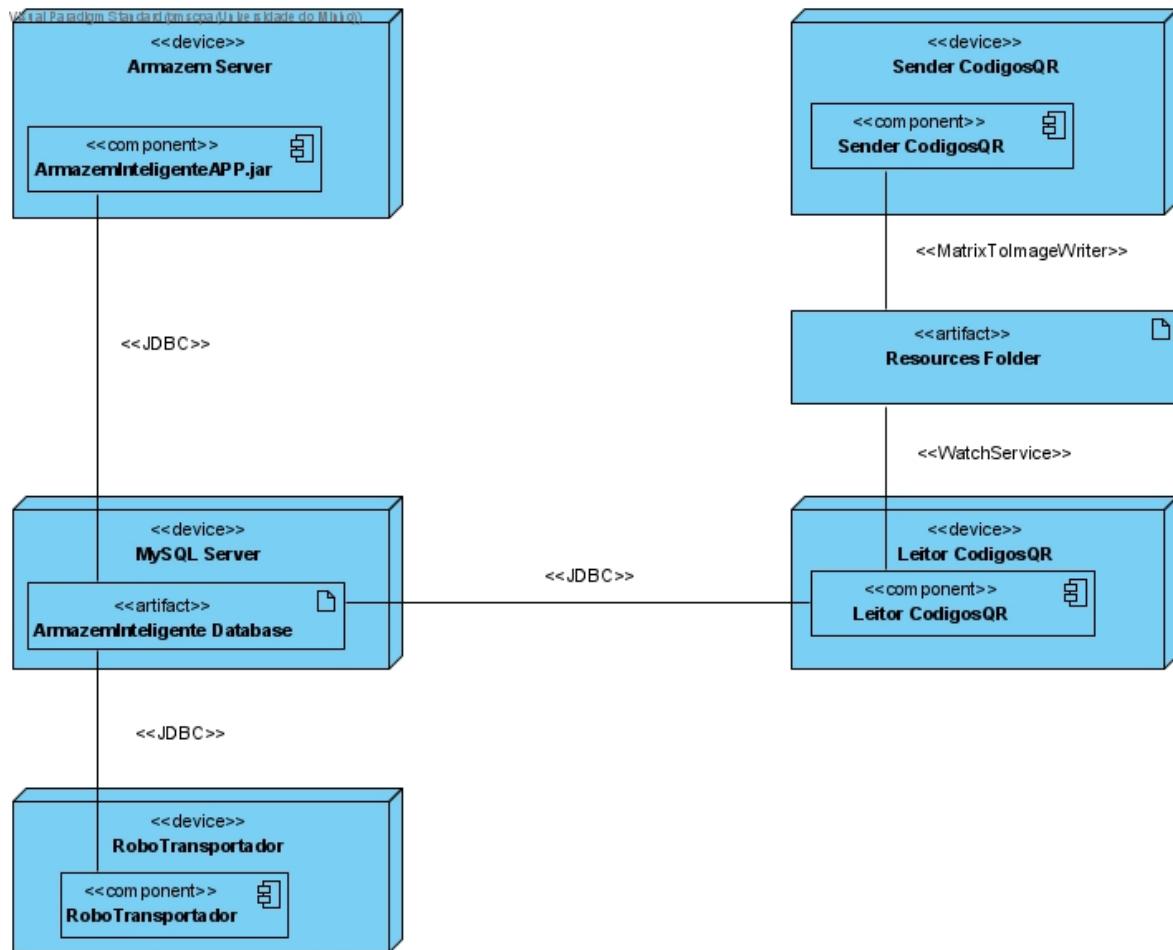


Figura 5.1: Diagrama de Instalação

O Armazém será composto por vários sistemas diferentes, que estabelecem canais de comunicação entre si, de forma a partilhar informação relevante a outros sistemas. Não haverá, porém, em momento algum, controlo de um dos sistemas por parte de outro, sendo que cada

um desconhece a implementação dos outros.

Os sistemas que irão compor o Armazém serão: o **Servidor Armazém**, a **Base de Dados (MySQL)**, o **Leitor de Códigos QR**, o **Sender de Códigos QR** e vários **Robôs de Transporte**.

## 5.1 Sender de códigos QR

Este sistema será um executável que irá receber do utilizador que material deve ser codificado para um Código QR e, efetivamente, gerar esse Código QR, através de um **MatrixToImageWriter**. Os códigos QR gerados serão guardados na diretoria **resources**.

## 5.2 Leitor de códigos QR

Este sistema irá interpretar cada Código QR que chega ao armazém. De forma a detetar a chegada de códigos, este sistema implementa um **Watch service** sobre a pasta **Resources**, pasta esta que é povoada pelo sistema anteriormente mencionado. O sistema irá descodificar estes códigos e inserir as Paletes que eles representam na Base de Dados **MySQL**.

## 5.3 Robo de transporte

Existirão várias instâncias deste sistema, cada uma sendo uma implementação física de um Robô do armazém. Este irá comunicar através de **Notificações** enviadas para a **Base de Dados**, direcionadas ao **Servidor do armazém**. Essas **Notificações** serão correspondentes ao término de cada uma das fases do transporte de Paletes.

Através do uso de notificações e da base de dados, o servidor não terá controlo direto sobre os robôs, espelhando o funcionamento real dos sistemas de armazenamento inteligente onde robôs se deslocam fisicamente, não sendo o seu movimento simulado pelo armazém, mas apenas a sua rota calculada e transmitida.

## 5.4 Servidor do Armazém

O **Servidor do armazém** configurará o componente principal do Armazém. Este servirá como entidade central de processamento da informação relativa ao funcionamento do armazém. Tanto a sua leitura, como o envio da sua informação, serão feitos exclusivamente através da **Base de Dados**, permitindo obter abstração da implementação dos outros sistemas.

Da **Base de Dados** irá ler tanto as novas Paletes chegadas ao sistema, inseridas pelo Leitor de Códigos QR, como as notificações enviadas pelos vários Robôs. Obtendo estas informações, alterará informação relativa às Paletes e Prateleiras. Calculará também rotas para os **Robos Transportadores** que irá transmitir através da **Base de Dados** de volta a estes sistemas, não havendo controlo direto sobre eles.

## Capítulo 6

# Implementação da Aplicação

Após todas as alterações mencionadas na secção anterior, começou-se assim a programar a aplicação pretendida, tendo novamente uma visão clara do funcionamento que queríamos implementar, e como o fazer.

A programação da aplicação, tal como esperado, deu-se sem incidentes, uma vez que através de um bom planeamento já sabíamos exatamente as interações que todos os seus componentes viriam a ter, e foi assim possível dividir a sua implementação em blocos modulares de modo a optimizar o processo.

O resultado é a aplicação que se encontra neste mesmo anexo, e que encorpora os **Use Cases** propostos da seguinte maneira:

### 6.1 Comunicar código QR

O Leitor de Códigos QR implementa um **Watch Service** sobre a pasta **resources**, onde chegarão Códigos QR gerados pelo **Sender de Códigos QR**. Quando um novo código é adicionado na pasta, é ativada uma chave de observação que assinala o nome do ficheiro adicionado, sendo este aberto com um descodificador de códigos, obtendo-se o material codificado.

É, de seguida, criada uma Paleta com o material descodificado, assinalando-se a Paleta como "Recém chegada" ao sistema. O Leitor de Códigos QR insere esta Paleta na **Base de Dados**, de forma a que o sistema posteriormente a atribua a um Robo.

O seguinte **Diagrama de Sequência**, representa o use case acima supramencionado:

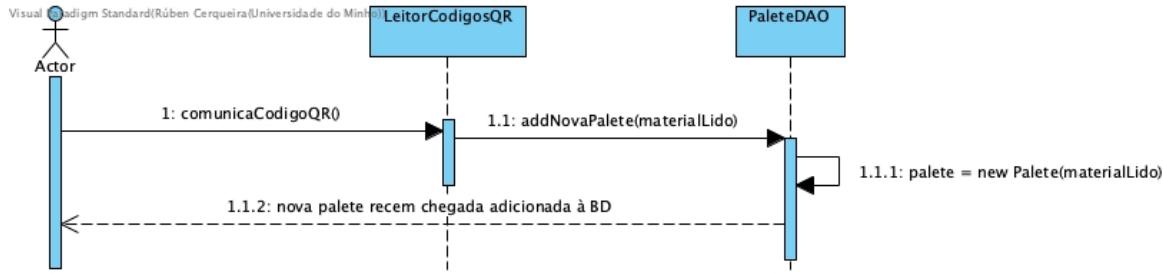


Figura 6.1: Diagrama de sequência - Comunicar código QR

## 6.2 Sistema comunica ordem de transporte

Para se escalonar uma palete, devem-se cumprir três requisitos:

- O armazém não deve estar cheio;
- Deve existir uma Paleta com o estado "Recém chegada";
- Deve existir um Robô livre.

Caso estas três condições se verifiquem, é procurado o id de uma Paleta recém chegada, e o id de uma Prateleira na qual se irá guardar a Paleta. Também é procurado o id e as coordenadas atuais do Robô que a deverá transportar.

O sistema calcula, nesse momento, a Rota desde a localização atual do Robô até à Prateleira escolhida.

É utilizado o DAO do Robo de forma a atualizar, no Robô, a informação relativa à Paleta que carrega, a Prateleira a que se destina e também para mudar o seu estado para "Em transporte".

A Rota é também inserida na **Base de Dados**, utilizando para tal o DAO respetivo.

Numa fase posterior ao **Use Case**, o Robô Transportador, externo ao servidor, irá ler da **Base de Dados** a Rota que deve seguir.

O seguinte **Diagrama de Sequência**, representa o **Use Case** supramencionado:

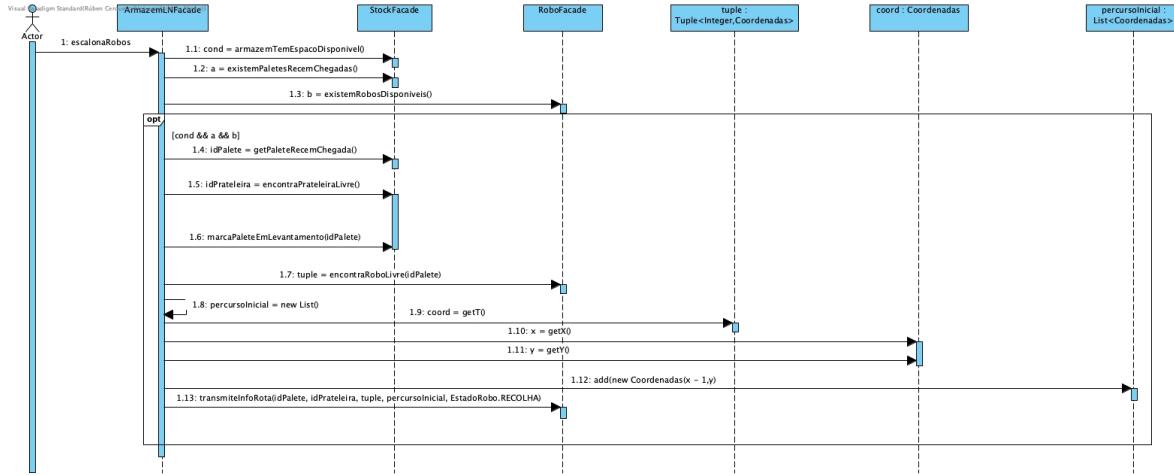


Figura 6.2: Diagrama de sequência - Sistema comunica ordem de transporte

### 6.3 Notificar recolha de Paletes

Quando o Robô termina a sua Rota atual e verifica que o seu estado se encontra em "Recolha", este enviará uma Notificação de recolha da Palete, direcionada ao sistema.

Visto o Robô Transportador ser uma entidade separada do sistema central, o comportamento do **Use Case** termina com o envio da Notificação. Julga-se, porém, importante explicitar o processamento dessa Notificação pelo sistema.

Aquando da leitura da notificação, o sistema irá marcar a Palete correspondente ao Robô como "Em transporte", assim como calcular uma Rota desde a posição atual do Robô até à posição da Prateleira reservada para a Palete. Esta Rota será posteriormente comunicada ao Robô Transportador, como explicitado no **Use Case** anterior.

O seguinte **Diagrama de Sequência**, representa o **Use Case** supramencionado:

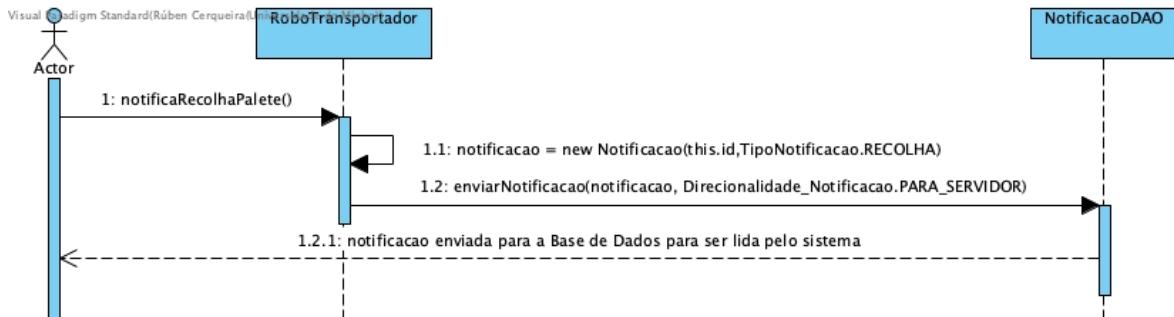


Figura 6.3: Diagrama de sequência - Notificar recolha de paletes

## 6.4 Notificar entrega de Paletes

O funcionamento deste **Use Case** será bastante semelhante ao anterior.

Quando o Robô termina a sua Rota atual e verifica que o seu estado se encontra em "Em Transporte", este enviará uma Notificação de entrega da Paleta, direcionada ao sistema.

Em semelhança ao **Use Case** da recolha das Paletes, a mecânica correspondente a este **Use Case** também termina com o envio da Notificação, porém será explicitado o seu processamento do lado do servidor.

No momento de leitura da Notificação de entrega, o sistema marca a Paleta como guardada e a Prateleira como estando ocupada. Calcula seguidamente a Rota de regresso do Robô, isto é, a Rota desde as suas Coordenadas atuais até às Coordenadas da zona de estacionamento do Robô, pré-definida pelo programa, indicando ao Robô para mudar o seu estado para "Regresso".

O seguinte **Diagrama de Sequência**, representa o **Use Case** supramencionado:

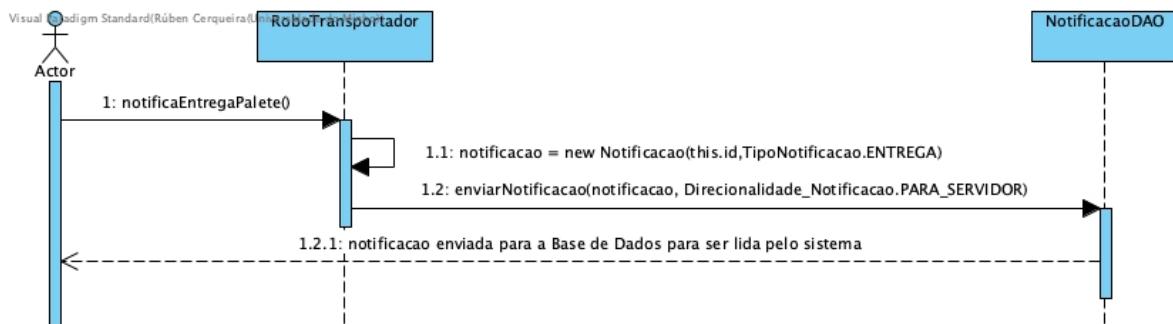


Figura 6.4: Diagrama de sequência - Notificar entrega de Paletes

## 6.5 Consultar listagem de localizações

Este é o último **Use Case** a considerar para a presente fase do projeto, e a sua implementação passa por o Gestor, quando devidamente autenticado, selecionar a segunda opção que lhe é oferecida.

É disponibilizada uma tabela com todas as Paletes no sistema, composta pelas seguintes colunas: ID da Paleta, Material contido nesta, Coordenadas da localização, e estado atual (Armazenada, em transporte, ou em espera).

O seguinte **Diagrama de Sequência**, representa o **Use Case** acima mencionado:

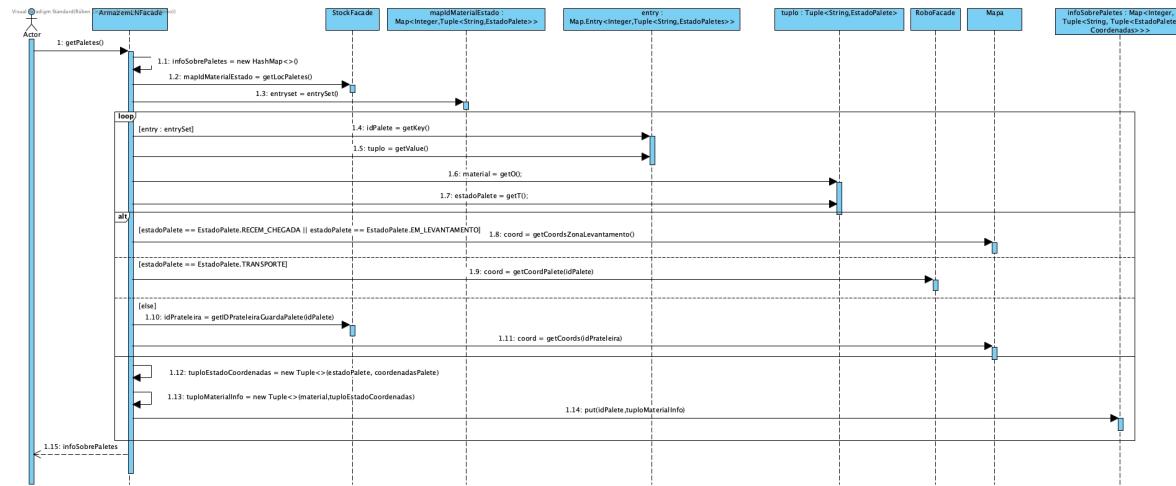


Figura 6.5: Diagrama de sequência - Consultar listagem de localizações

Embora não se inclua num **Use Case**, consideramos pertinente incluir um **Diagrama de Sequência** adicional:

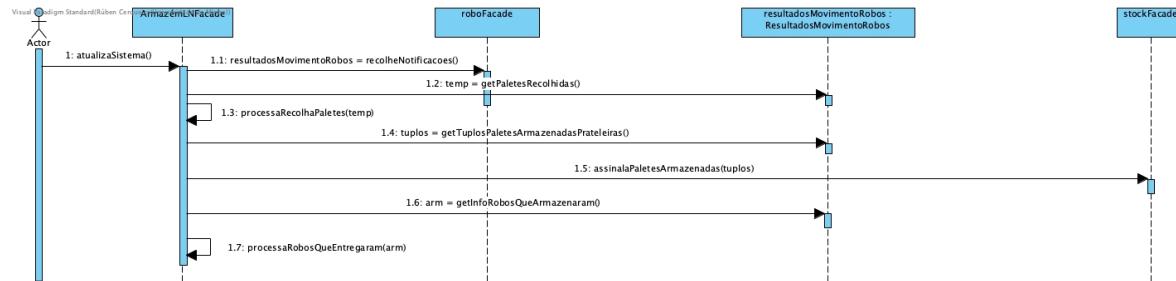


Figura 6.6: Diagrama de Sequência atualizaSistema

Este descreve o procedimento que é adoptado no final de cada unidade de tempo decorrida no sistema, e garante que todas as Paletes são devidamente atualizadas assim que são armazenadas numa Prateleira.

# Capítulo 7

## Análise Crítica

Fazendo uma análise crítica da aplicação desenvolvida, e porque não há aplicação perfeita, cremos que ainda há pontos em que esta poderia ser melhorada, dos quais se destacam:

- As Rotas poderiam estar a ser criadas evitando choques com outros Robos.
- A interface com o utilizador podia estar implementada com recurso a uma GUI ao invés de estar simplesmente em formato de texto
- A implementação tem ainda a possibilidade de incorporar as especificações originais da terceira fase, incluindo o armazenamento de matérias perecíveis e matérias não perecíveis, assim como o tratamento de pedidos de requisição para envio de matérias para fora do armazém.
- De momento, apenas uma Paleta é escalonada em cada unidade de tempo, isto é, se chegarem, num momento, 3 Paletes novas ao sistema, serão necessárias 3 unidades temporais de forma a conseguir reservar um Robo e uma Prateleira correspondente ao transporte e armazenamento de cada Paleta. A melhoria proposta consistiria no escalonamento de todas as Paletes recém chegadas numa unidade de tempo apenas, o que permitiria atenuar a formação de bottlenecks e filas de espera de Paletes a aguardar escalonamento.

Citando os **Uses Cases** de Notificar Recolha de Paletes e Notificar Entrega de Paletes, uma crítica que se poderá apontar, será a um aspeto que julgamos importante salientar, relacionado com o controlo de concorrência. Este, é algo fulcral ao bom funcionamento do projeto, visto o armazém ser composto por vários componentes independentes entre si, cada um controlado por Threads diferentes. Estes enumeram-se do seguinte modo: a Thread principal, que interage com o utilizador, a Thread responsável pelo leitor de código QR, cada uma das Threads que controlam um dos Robôs e a Thread responsável por escalar Robos, Paletes e processar as Notificações por eles enviadas.

Foram feitos esforços na medida de eliminar situações onde possam surgir problemas de concorrência, porém, dado o limitado conhecimento relativo à área dos Sistemas Distribuídos,

reconhece-se a possibilidade de surgirem situações que levem ao mau funcionamento do sistema. Como tal, é proposta uma nova versão do projeto, em que a forma de comunicação entre Robôs e sistema não é feita através da **Base de Dados**, mas sim através de **Sockets** numa arquitetura cliente-servidor, onde o fluxo de informação é mais fiável e facilmente controlável.

Esta seria uma implementação superior do modelo de comunicação, e algo que gostaríamos de implementar no futuro.

# Capítulo 8

## Conclusão

Com esta fase concluímos o processo de desenvolvimento da aplicação Armazém Inteligente.

Sendo o final de um processo iterativo, consideramos pertinente falar do projeto como um todo, e não desta fase num vácuo.

Através do desenvolvimento da aplicação, aprendemos como estipular um plano faseado, em que cada fase subsequente assenta diretamente sobre a anterior. Houve inevitavelmente algumas alterações de fase para fase, mas com um foco completamente iterativo, que assegura que não temos que reformular todo o nosso código ao nos apercebermos que existem ligações que não haviam sido consideradas, ou componentes em falta.

Começámos por fazer um estudo pormenorizado sobre o funcionamento de um armazém real, percebendo a interação de cada um dos elementos, assim como as particularidades inerentes ao sector.

Deste estudo foram desenvolvidos o **Modelo de Domínio** e os **Diagramas de Use Cases**. Através destes, é já possível ter uma noção geral das entidades envolvidas, assim como os processos que levam ao funcionamento expectável de um armazém.

A partir destes, elaboraram-se **Diagrama de Componentes**, o **Diagrama de Classes**, e os **Diagramas de Sequência**. Estes já têm muito mais pormenor e especificidade relativa ao projeto. Temos uma clara noção da maneira como cada entidade será representada, assim como os métodos que levam à modelação da funcionalidade desejada.

Nesta última fase, após as devidas reformulações aos diagramas anteriores, fruto da mudança de especificações e de uma melhor formulação da funcionalidade pretendida, implementámos todo o planeamento desenvolvido ao longo das duas primeiras fases na forma de uma aplicação.

Para tal, foi necessário definir um novo tipo de diagrama, o **Diagrama de Máquina de Estados**, para descrever qual será a interação que o utilizador pode ter com a aplicação quando lhe acede, e os diferentes estados que esta pode tomar.

Durante a implementação da aplicação, uma vez que foram usados dois sistemas externos para a comunicação com o Leitor de Códigos QR e os Robôs, houve necessidade ainda de definir um último tipo de diagrama, o **Diagrama de Instalação**. Este permite ter noção da maneira como os diferentes sistemas interagem para que a aplicação produza os resultados desejados.

Com a conclusão da aplicação, dá-se também por terminado todo o processo de desenvolvimento, em que se seguiram todas as estipulações e requisitos para esta fase, mesmo havendo ainda algumas facetas da aplicação que podem ser melhoradas e funcionalidades adicionais a implementar, tal como falado na secção anterior. Trabalhou-se de forma completamente faseada, o que leva a um resultado íntegro, que corresponde ao planeamento que foi feito no decorrer das fases anteriores, e se traduz numa aplicação com a qual estaríamos satisfeitos em apresentar ao cliente.