Universidade do Minho
Mestrado Integrado em Engenharia Informática

DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE
FASE 3
GRUPO 11

André Carvalho da Cunha Martins a89586
António Jorge Nande Rodrigues A89585
José Pedro Castro Ferreira A89572
Rui Emanuel Gomes Vieira A89564

23 de dezembro de 2020


A89586


A89585


A89572


A89564

Conteúdo

1 Introdução

Nesta última fase do trabalho prático, passamos à construção da nossa aplicação. Após termos analisado e definido os Use Cases nas fases anteriores, nesta fase foram implementados. Os Use Cases implementados nesta fase foram **Gestor inicia sessão**, **Leitor comunica código QR**, **Gestor solicita localização das paletes**, **Notifica recolha de palete** e **Notifica entrega de palete**. Ao realizarmos a implementação, e à medida que íamos desenvolvendo a nossa aplicação, reparamos que era necessário efetuar alguns reparos aos Use Cases, uma vez que a implementação foi feito de uma maneira ligeiramente diferente. Com este trabalho prático abordamos, pela primeira vez, a definição de classes DAO(Data Acess Object) e o uso de uma base de dados, em vez da gracação de dados em memória ou em ficheiro de texto. No nosso caso, utilizamos o SGBD MySQL (principalmente pela pática com esta ferramento, utilizada na UC Bases de Dados). Definimos o modelo lógico da nossa base de dados e passamos para a usa implementação física e posterior povoamento. O uso de DAO foi algo muito interessante e bastante prático, uma vez que os nossos dados ficavam sempre guardados numa base de dados, não sendo necessário estar sempre a colocar os dados em memória ou a ler um ficheiro de texto, como haviamos feito em UC's anteriores.

2 Use Cases

Como foi dito em cima, nesta fase implementamos 6 Use Cases, tendo efetuado algumas alterações em relação à maneira como estavam definidos nas fases 1 e 2. De seguida, apresentamos a nova definição desses mesmos Use Cases.

2.1 Use Case: Gestor inicia sessão

Cenário: Cenário 3
Pré-condição: True
Pós-condição: Gestor fica com sessão iniciada no sistema **Fluxo Normal:**
1. Sistema pede código de acesso para o gestor realizar o login
2. Gestor insere código de acesso
3. Sistema valida credenciais e o login é efetuado

Fluxo alternativo 1: Gestor insere credenciais erradas (Passo 2)
1. Sistema informa que credenciais estão incorretas
2. Gestor insere de novo o código de acesso
3. Sistema valida credenciais e o login é efetuado com sucesso

Fluxo exceção: Gestor não possui credenciais corretas (Passo 2)
1. Sistema informa que credenciais estão incorretas
2. Gestor desiste de tentar iniciar sessão no sistema

2.2 Use Case: Leitor Comunica Código QR

Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete e QRCode são registados no sistema
Fluxo Normal:
1. Leitor analisa código QR de uma palete.
2. Sistema valida QR code.
3. Palete é registada no sistema.

Fluxo alternativo 1: Código QR é inválido (Passo 2)
1. Sistema indica que Código QR não é válido.
2. Leitor insere de um código válido.
3. Palete é registada no sistema.

2.3 Use Case: Gestor solicita localização das paletes

Cenário: Cenário 3
Pré-condição: True
Pós-condição: Gestor obtém listagem completa das paletes e da sua localização
Fluxo Normal:
1. Gestor envia pedido ao sistema da listagem de todas as paletes
2. Sistema consulta a localização das paletes
3. Sistema emite a listagem completa

2.4 Use Case: Notificação de transporte

Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete é transportada para uma prateleira ou zona de carga
Fluxo Normal:
1. Recolha de palete (zona de descarga ou prateleira) é aprovada
2. Robot transporta palete até prateleira ou zona de carga
3. Sistema atualiza localização da palete

2.5 Use Case: Notifica recolha de palete

Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete é transportada para uma prateleira
Fluxo Normal:
1. Sistema envia pedido para recolher palete da zona de carga

2. Robot disponível vem recolher palete
3. Robot marcado como indisponível
4. Palete transportada para uma prateleira

Fluxo de exceção 1: Armazém cheio (passo 4)
4.1. Sistema é informado que o armazém está cheio
4.2. Palete fica em espera até haver uma prateleira disponível

Fluxo de exceção 2: Robots indisponíveis (passo 2)
2.1 Sistema é informado que não há robots disponíveis
2.2 Palete fica em espera até haver um robot disponível

2.6 Use Case: Notifica entrega de palete

Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete é transportada para zona de carga
Fluxo Normal:
1. Sistema envia pedido para recolher palete de prateleira
2. Robot disponível vem recolher palete
3. Robot marcado como indisponível
4. Palete transportada para zona de carga

Fluxo de exceção 1: Não há paletes armazenadas (passo 1)
1.1. Sistema é informado que não há paletes nas prateleiras
1.2. Sistema cancela pedido

Fluxo de exceção 2: Robots indisponíveis (passo 2)
2.1 Sistema é informado que não há robots disponíveis
2.2 Palete fica em espera até haver um robot disponível

3 Diagrama de Classes e Diagrama ORM

Na fase 2, tínhamos idealizado um diagrama de classes. Ao definir a nossa aplicação, reparamos que era necessário reformular algumas classes, uma vez que o que nos foi pedido nesta fase 3 não distinguia palete perecível de não perecível, por exemplo. Abandonamos, também, a definição de Queues e a existência de uma superclasse Funcionário, uma vez que não se revelaram necessárias nesta fase do trabalho. Apresentamos, de seguida, as classes definidas e as suas variáveis de instância.

```
public class ArmazemFacade implements IArmazemFacade{
    private Map<Integer, Palete> paletes;
    private Map<String, Gestor> gestores;
    private Map<Integer, Robot> robots;
    private int[] corredor1 = new int[5];
    private int[] corredor2 = new int[5];
```

```java
public class Gestor {
    private String password;
    private String nome;
```

```java
public class LeitorQR {

    private Map<Integer, Palete> paleteRegistada;
```

```java
public class Localizacao {
    private int zona;
    private int prateleira;
```

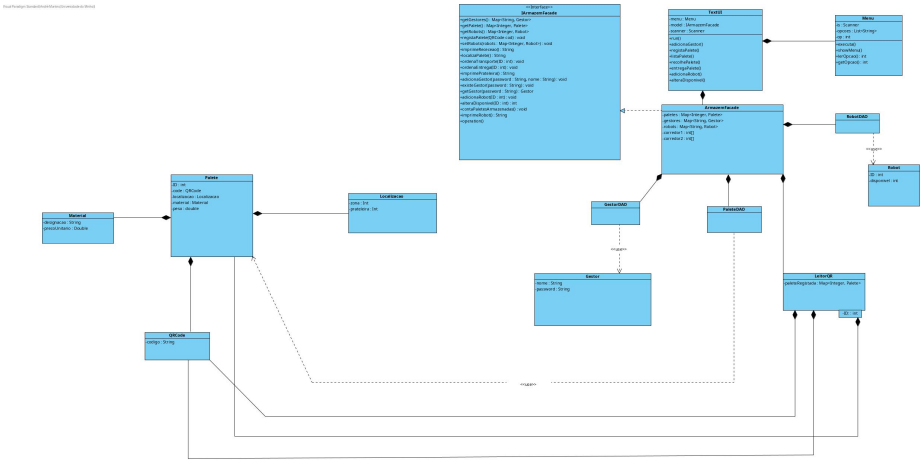
```java
public class Material {
    private String designacao;
    private double precoUnitario;
```

```java
public class Palete {
    private int ID;
    private QRCode code;
    private Localizacao localizacao;
    private Material material;
    private double peso;
```

```java
public class QRCode {
    private String codigo;
```

```java
public class Robot {
    private int id;
    private int disponivel;
```

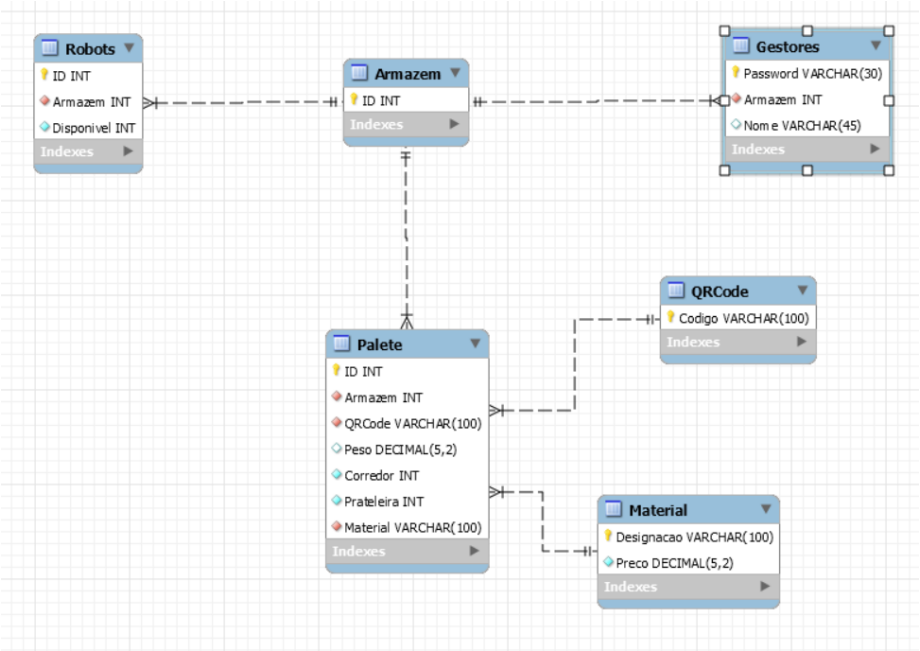
De seguida, apresentamos o Diagrama ORM da nossa aplicação, onde está representadas todas as nossas classes, bem como os DAO utilizados. Este diagrama foi baseado no diagrama de classes da fase 2, tendo sofrido algumas alterações nesta última fase do trabalho prático (Uma vez que se encontra um pouco impercetível, iremos enviar o ficheiro.jpg como anexo)



4 Implementação da Base de Dados

Ao realizar a implementação e construção da nossa aplicação, foi-nos proposto o uso de uma base de dados para armazenar os nossos dados e classes DAO para gerir essa mesmo base de dados. As classes DAO implementavam uma Map e, a partir da definição de cada método inerente a um Map, foi nos possível manipular e gerir a nossa base de dados, sendo possível, adicionar, remover e atualizar todos os dados lá presentes.
A nossa base de dados, denominada **Armazem**, possui um total de 6 relações, sendo elas **Armazem**, **Gestores**, **Material**, **Palete**, **QRCode** e **Robots.**
Com a ajuda de uma biblioteca específica para realizar uma conexão a uma base de dados (Neste caso, MySQL) através de código Java, construímos uma base de dados com o modelo lógico que iremos apresentar de seguida, bem como a definição de cada uma das tabelas

5 Modelo Lógico



6 Definição das Tabelas

```
mysql> show tables;
+--------------------+
| Tables_in_armazem  |
+--------------------+
| armazem            |
| gestores           |
| material           |
| palete             |
| qrcode             |
| robots             |
+--------------------+
6 rows in set (0.00 sec)

mysql> _
```

6.1 Armazem

```
mysql> describe Armazem;
+-------+------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+-------+------+------+-----+---------+-------+
| ID    | int  | NO   | PRI | NULL    |       |
+-------+------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

6.2 Gestores

```
mysql> describe Gestores;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Password | varchar(30) | NO   | PRI | NULL    |       |
| Armazem  | int         | NO   | MUL | NULL    |       |
| Nome     | varchar(45) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

6.3 Material

```
mysql> describe Material;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| Designacao | varchar(100) | NO   | PRI | NULL    |       |
| Preco      | decimal(5,2) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

6.4 Palete

```
mysql> describe Palete;
+------------+--------------+------+-----+---------+-------+
| Field      | Type         | Null | Key | Default | Extra |
+------------+--------------+------+-----+---------+-------+
| ID         | int          | NO   | PRI | NULL    |       |
| Armazem    | int          | NO   | MUL | NULL    |       |
| QRCode     | varchar(100) | NO   | MUL | NULL    |       |
| Peso       | decimal(5,2) | YES  |     | NULL    |       |
| Corredor   | int          | NO   |     | NULL    |       |
| Prateleira | int          | NO   |     | NULL    |       |
| Material   | varchar(100) | NO   | MUL | NULL    |       |
+------------+--------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

6.5 QRCode

```
mysql> describe QRCoDE;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| Codigo | varchar(100) | NO   | PRI | NULL    |       |
+--------+-------------+------+-----+---------+-------+
1 row in set (0.00 sec)
```

6.6 Robots

```
mysql> describe robots;
+------------+------+------+-----+---------+-------+
| Field      | Type | Null | Key | Default | Extra |
+------------+------+------+-----+---------+-------+
| ID         | int  | NO   | PRI | NULL    |       |
| Armazem    | int  | NO   | MUL | NULL    |       |
| Disponivel | int  | YES  |     | 1       |       |
+------------+------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

7 Povoamento da Base de Dados

Para a nossa aplicação ser funcional, era necessário povoar a nossa base de dados. Definimos 3 SQL scripts para povoar a base de dados com todos os dados necessário e, também, acrescentamos mais dados já através da nossa aplicação, testanto, assim a conexão entre o nosso programa e a nossa base de dados, sendo esssa conexão bem sucedida.

```
mysql> select * from palete;
+----+---------+--------------------+-------+----------+------------+----------+
| ID | Armazem | QRCode             | Peso  | Corredor | Prateleira | Material |
+----+---------+--------------------+-------+----------+------------+----------+
|  1 |       1 | 1&&Fruta&&5&&6     |  6.00 |        0 |          0 | Fruta    |
|  2 |       1 | 2&&Pregos&&1&&3    |  3.00 |        1 |          1 | Pregos   |
|  3 |       1 | 3&&Polvo&&5&&5     |  5.00 |        2 |          4 | Polvo    |
|  4 |       1 | 4&&Carne&&10&&6    |  6.00 |       -1 |         -1 | Carne    |
|  5 |       1 | 5&&Cerveja&&2.5&&40 | 40.00 |        1 |          3 | Cerveja  |
+----+---------+--------------------+-------+----------+------------+----------+
5 rows in set (0.00 sec)
```

```
mysql> select * from robots;
+----+---------+------------+
| ID | Armazem | Disponivel |
+----+---------+------------+
|  0 |       1 |          0 |
|  1 |       1 |          0 |
|  2 |       1 |          0 |
|  3 |       1 |          0 |
|  4 |       1 |          1 |
+----+---------+------------+
5 rows in set (0.00 sec)
```

```
mysql> select * from gestores;
+----------+---------+----------+
| Password | Armazem | Nome     |
+----------+---------+----------+
| 12345    |       1 | Manuel   |
| 56789    |       1 | André    |
| 67890    |       1 | João     |
| teste123 |       1 | António  |
+----------+---------+----------+
4 rows in set (0.00 sec)
```

8 Funcionalidades da Aplicação

Definimos 7 funcionalidades para a nossa aplicação, sendo elas **Comunicar QRCode**, **Notificar recolha de palete**, **Notificar entrega de palete**, **Consultar listagem de palete**, **Registar Novo Gestor**, **Adicionar Robot**, **Alterar disponibilidade de Robot**.

8.1 Comunicar QRCode

Esta funcionalidade tem algumas especificações. Aqui, é inserio um código QR relativo a uma palete. Esse código contém informação sobre o ID da palete, material que a mesma transporta, preço unitário desse material e peso total da palete. Contudo, o QRCode tem uma forma muito específica de ser escrito que é a seguinte `ID&&Material&&PrecoUnitario&&Peso`. Qualquer input que não seja desta forma é marcado como inválido não sendo, assim, possível o registo de uma nova palete.

8.2 Notificar recolha de palete

Nesta funcionalidade, é dado como input o ID, válido, da palete que se pretende transportar para um palete. Depois, é analisado se existem espaço no armazém e se existem robots disponíveis. Se estas duas condições se verificarem, a localização da palete é atualizada e o robot que a transportou é marcado como indisponível.

8.3 Notificar entrega de palete

À semelhança da recolha de palete, esta funcionalidade verifica se o ID é válido, se existem paletes em prateleiras e se existem robots disponíveis. Novamente, o robot que efetuou o transporte é marcado como indisponível.

8.4 Consultar Listagem Palete

É nesta funcionalidade que é necessário verificar o login de um gestor. Se o código de acesso for válido, o gestor tem acesso à localização de todas as paletes, bem como o seu peso e preço total da palete.
Nota: A localização 0 0 é marcada como zona de descarga (Receção) e a localização -1 -1 é marcada como zona de carga (Zona de Entrega)

8.5 Registar novo gestor

Nesta funcionalidade efetua-se o registo de um novo Gestor. Primeiro, pede se para inserir um código de acesso que identifique esse gestor (Único e que ainda não esteja registado na base de dados). De seguida, pedimos para se inserir o nome e um novo gestor é registado na base de dados.

8.6 Adicionar um novo Robot

Esta funcionalidade é muito semelhante à adição de um novo gestor. Pedimos para inserir um ID para o novo robot, único e que ainda não tenha sido registado. Depois é registado um novo robot na base de dados, sempre marcado como disponível.

8.7 Alterar disponibilidade de Robot

Esta funcionalidade revelou-se bastante importante, apesar da sua simplicidade. Uma vez que, sempre que um robot efetua um transporte, é marcado como indisponível, é necessário marca-lo como disponível novamente. Esta funcionalidade também pode funcionar no sentido inverso, existindo um robot marcado como disponível mas que é necessário inativar (Por avaria ou outros fatores externos). Após a alteração, a disponibilidade do robot é alterada na base de dados.

9 Análise crítica de resultados obtidos, utilizando Use Case: Leitor comunica Código QR

Na primeira fase identificamos este Use Case de uma maneira muito diferente daquela que viria a ser implementada. Tanto na fase 1, como na fase 2 também, existia a possibilidade do sistema gerar um novo QRCode. Tal não foi definido nem implementado na última, uma vez que entendemos que o QRCode é algo inerente a uma palete, e não faria sentido o nosso sistema estar a gerar novos QRCode sempre que algum era dado como inválido. Assim, em vez de se gerar esse novo QRCode, damos a oportunidade de inserir de novo o QRCode. No entanto, o cerne do Use Case foi sempre seguido, podendo dividir este em 3 fases.

1. Leitura
2. Validação
3. Registo

Estes 3 passos são seguidos nas 3 fases do projeto. Na primeira fase tínhamos uma visão muito mais complexa da aplicação que iríamos construir, havendo separação entre carga e descarga de palete, ambas associadas à leitura de QR-Codes. Na segunda fase, já houve uma simplificação significativa do que a nossa aplicação teria que fazer, tendo, na última fase, sido feito o único reparo de não se gerar um novo QRCode.

De um modo mais geral, e analisando todo o desenvolvimento deste trabalho prático, consideramos que a nossa ideia da aplicação a desenvolver começou como sendo algo muito complexo e passou para algo mais simples e eficiente. Isto mostra que a primeira idealização da aplicação não é inerte e pode sofrer várias alterações à medida que o projeto é desenvolvido.

Consideramos, também, que é só na última fase do trabalho prático que tivemos a noção completa do que seria necessário fazer e de que maneira teria que ser feito. As duas fases anteriores foram bastante utéis para idealizar e ter uma ideia geral da nossa aplicação, de certa forma, construimos o esquelo daquilo que é o nosso programa. Quando chegou a altura da implementação, já tínhamos uma noção geral de como se iria comportar o nosso programa, o que teria que fazer e de que maneira iria responder, realizando os ajustes necessários para que este funcionasse da melhor maneira possível.

10 Conclusão

Dado por concluída a última fase do trabalho prático, iremos fazer um balanço geral do projeto.
Na primeira fase, idealizamos e ficamos com uma visão mais geral, e bastante mais complexa, daquilo que iria ser a nossa aplicação a ser desenvolvida, com muito mais funcionalidades e entidades do que aquelas que chegaram a ser implementadas.
Na segunda fase, e com a análise de apenas 8 Use Cases, percebemos que era necessário redefinir os Use Case previamente definidos na fase 1. Após isso, formamos um diagrama de classes, com aqueles que pareciam, à altura, serem as classes necessárias para o desenvolvimento do nosso projeto. Ao desenvolver os diagramas de sequência, tentamos definir a melhor forma do nosso programa responder ao que lhe era pedido.
Foi só na terceira, e última, fase do nosso projeto que nos apercebemos realmente de como o nosso programa iria funcionar na íntegra. Contudo, as duas fases anteriores e o seu estudo permitiram uma implementação muito mais rápida e eficiente do que se tívessemos começado do 0. Ainda que os modelos definidos não tenham sido seguidos a 100%, a sua definição ajudou muito na elaboração do nosso programa, uma vez que já tínhamos uma ideia e, de algum modo, o "esqueleto"da aplicação e sabíamos de que maneira teriámos que definir os diferentes processos. Portanto, a definição de modelos é bastante útil na fase de implementação, dado que nos permite partir para a construção da aplicação já com uma ideia base, sendo este um ponto importante para a elaboração de uma aplicação mais competente, eficiente e que responda a tudo que lhe é pedido.
Para terminar, falemos um pouco da última fase em si, a fase onde "pusemos a mão na massa". Consideramos este fase bastante enriquecedora e interessante, uma vez que foi a primeira vez que definimos um programa que guarda os seus dados numa base de dados, e não me memória. O uso de DAO's revelou-se bastante útil e prático, uma vez que os nossos dados ficariam sempre guardados, independentemente do estado da nossa aplicação. Este novo método de trabalho é bastante interessante e será, sem dúvida, algo a ser utilizado na construção de aplicações futuras.