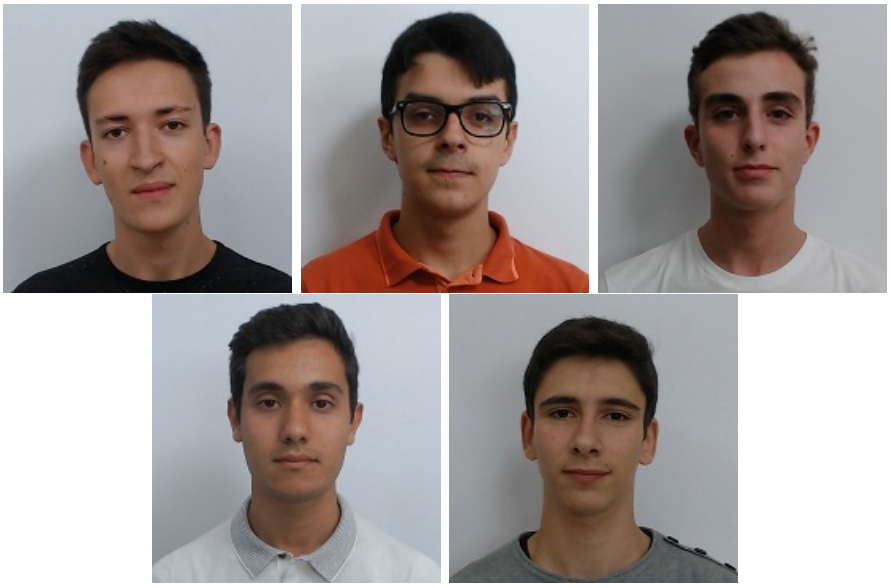
Desenvolvimento de Sistemas de Software
Fase 3
Grupo 34

Benjamim Coelho, Henrique Neto, Júlio Alves
Paulo Pereira, Simão Monteiro

e-mail: {a89616,a89618,a89468,86475,85489}@alunos.uminho.pt

Dezembro 2020



Conteúdo

1 Introdução

O objetivo desta fase é implementar o processo de transporte das paletes, desde que o código QR é lido, até que as paletes são colocadas nas prateleiras. Para tal iremos basear-nos no trabalho realizado na fase 2. Por exemplo, apesar de o mapa sugerido pelos professores ser bastante mais simples do que a nossa ideia, utilizámo-la na mesma, para ficar de acordo com os diagramas e planeamento que fizemos na fase anterior.

Para além disto, precisamos de implementar pelo menos uma vista para o programa. Neste caso, uma interface simples em modo texto é o suficiente.

Por último, e de modo a haver persistência de dados no nosso programa, iremos proceder à utilização do sistema de gestão de base de dados MySQL.

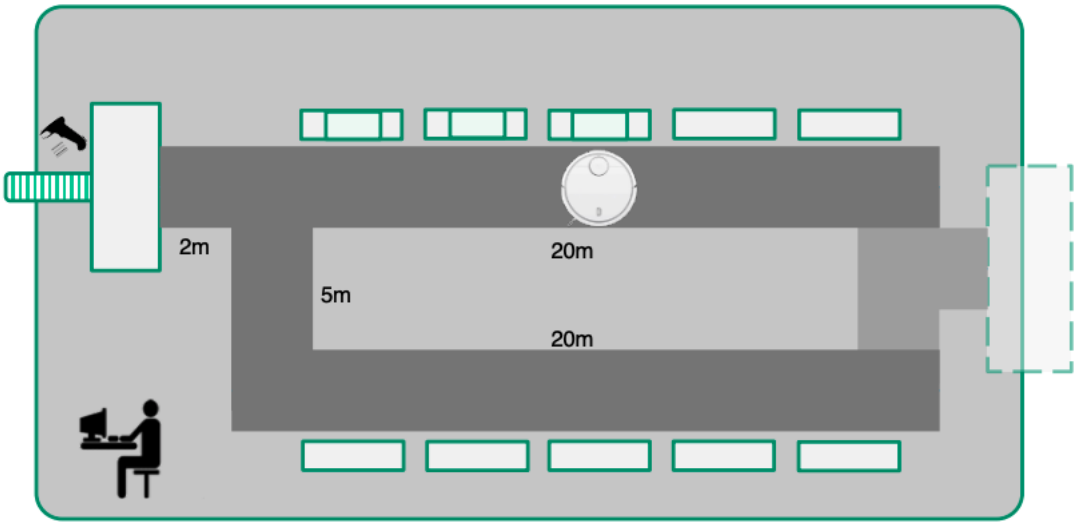
2 Use Cases a implementar

Use Cases:

- 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)

2.1 Simulação do processo de transporte das paletes

2.1.1 Mapa do armazém



O armazém consiste em dois corredores paralelos de 20 metros de comprimento com 5 prateleiras cada, distanciados de 5 metros.

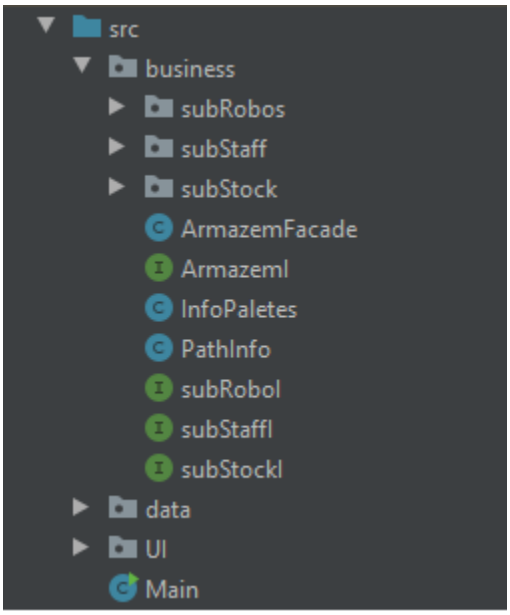
Será apenas considerado o caso da chegada de paletes ao armazém, pelo que deverá respeitar a seguinte ordem cronológica:

1. Motorista solicita autorização de descarga
2. Gestor autoriza pedido de descarga
3. Leitor de código QR comunica código da palete
4. Sistema notifica robô para transportar palete
5. Robô notifica recolha da palete
6. Robô informa sistema que realizou entrega com sucesso

3 Lógica de Negócio

Inicialmente, e de acordo com o nosso diagrama de componentes realizado na fase anterior, dividímos o nosso projeto em 3 packages: o package UI irá conter todos os ficheiros relacionados com vistas para o nosso programa, o package business terá todos os ficheiros que estão relacionados com a nossa lógica de negócio e o package data que conterá os ficheiros que estão relacionados com a conexão do nosso programa ao sistema de gestão de base de dados.

Como planeámos na fase 2, a nossa lógica de negócio divide-se em 3 subsistemas/packages, o subRobos, o subStaff e o subStock. De seguida iremos entrar em mais detalhe sobre os mesmos.



3.1 SubRobos

Tal como foi referido na fase anterior, este subsistema é responsável por todas as ações que influenciam os robôs do nosso armazém.

3.2 SubStaff

Este subsistema, tal como explicado na fase anterior, é responsável por todas as funcionalidades que envolvem pessoas que trabalham no nosso armazém, por exemplo a sua autenticação.

Como nesta fase não foi necessária a implementação de todos os use cases, não houve necessidade de utilizar este subsistema. No entanto, implementámos uma parte para termos uma ideia de como o nosso programa poderia escalar.

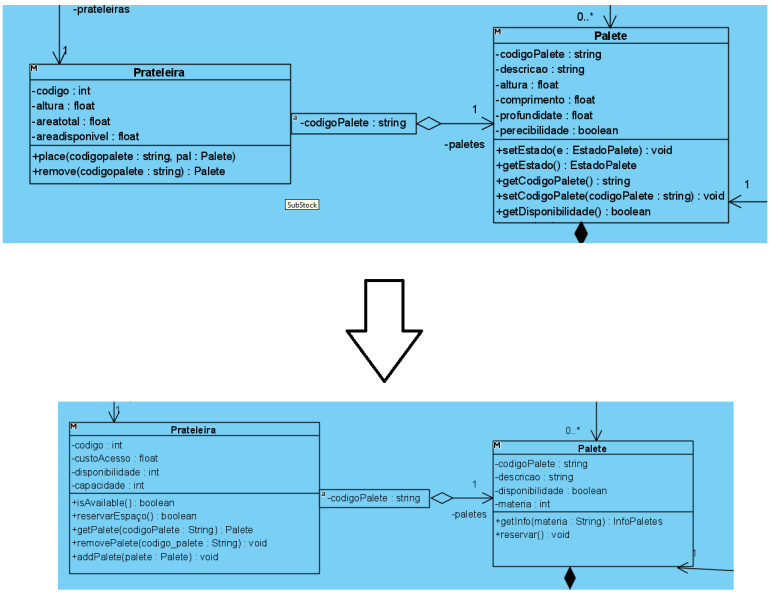
3.3 SubRequisicao

Como os use cases que envolvem este subsistema não foram selecionados para esta fase, não está presente na nossa implementação física.

3.4 SubStock

O subsistema SubStock é responsável por tudo que esteja relacionado com a logística de armazenamento de paletes no nosso armazém.

Uma pequena mudança que fizemos no nosso programa em relação à fase 2 foi a maneira como controlámos a quantidade de paletes que estão armazenadas numa prateleira. Na fase anterior tínhamos, tanto na prateleira como na palete, variáveis que correspondiam às dimensões do dado objeto (comprimento, altura e profundidade). Porém, nesta fase, apercebemo-nos que era uma ideia demasiado complexa e não seria vantajosa para a implementação pedida pela equipa docente para esta fase. Assim, na prateleira, temos um inteiro que representa a sua capacidade de armazenar paletes, tal como um inteiro que representa o número de espaços não reservados para paletes.



Como nesta fase não é necessário diferenciar zonas refrigeradas e não refrigeradas, uma vez que também não há distinção entre paletes com materiais perecíveis e paletes com materiais não perecíveis, não precisámos de implementar as classes ZonaRefrigerada e ZonaNaoRefrigerada.

4 Base de Dados

Se guardássemos as informações relativas ao programa exclusivamente em memória (como temos feito em projetos anteriores), quando o terminássemos perderíamos essa mesma informação. Este problema leva à necessidade de se utilizar bases de dados que podem armazenar dados de forma segura e permanente.

4.1 Sistema de gestão de bases de dados

Assim, de modo a obter persistência de dados utilizámos o sistema de gestão de base de dados MySQL pelas seguintes razões:

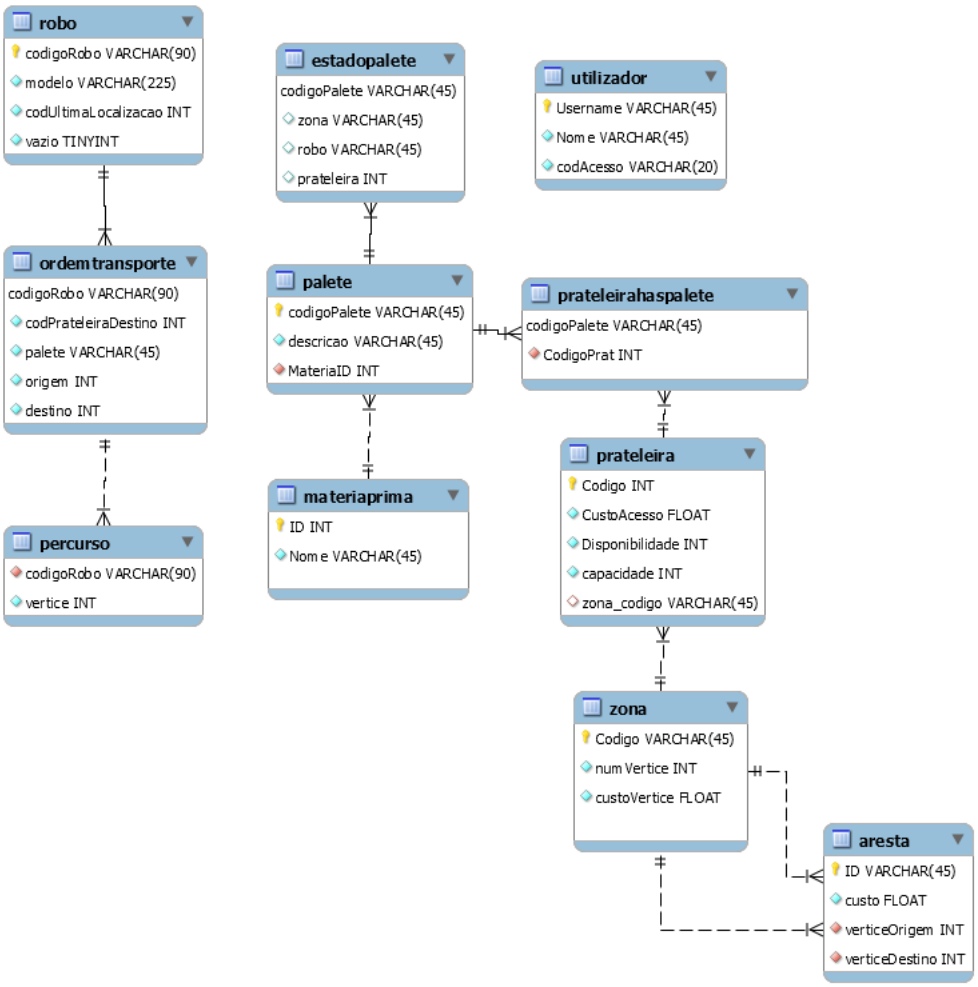
- Foi o sistema utilizado durante as aulas de Bases de Dados neste semestre, o que nos facilita esta fase, uma vez que é o sistema em que temos mais experiência de utilização e referências.
- Permite a criação de vários perfis de utilizador, o que suportará os nossos requisitos de controlo, isto é, que funcionalidades é que certos tipos de utilizadores irão ter acesso.
- É um sistema seguro e economicamente acessível comparadamente a outras soluções no mercado.
- A funcionalidade de backups proporciona um alto nível de segurança e confiança.
- Tem um alto desempenho, desde o baixíssimo tempo que demora desde fazer download até à instalação e configuração até ao rápido tempo de reposta a tarefas como queries, procedures...
- O MySQL proporciona também um alto nível de escabilidade, na medida em que é bastante flexível caso seja necessário modificar uma certa base de dados para dar suporte a novas funcionalidades ou requisitos

Para estabelecer a ligação entre o Java e a nossa Base de Dados, precisámos de utilizar o connector do MySQL para o Java, disponibilizado gratuitamente no site do MySQL.

4.2 Data Access Objects

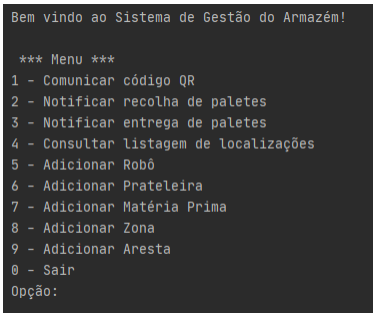
Implementar persistência de dados num programa consiste em criar Data Access Objects, aos quais iremos referir-nos como DAO's por uma questão de simplificação. Deste modo, os maps que tínhamos no nosso diagrama de classes passariam a ser DAO's que implementam um map. De forma a aceder à Base de Dados, criámos uma classe (*DAOConfig*) que possui todas as informações necessárias para aceder à base de dados, nomeadamente o username, a password, o URL e o driver que vamos utilizar. É,também, nesta classe que criamos a conexão à base de dados, de forma a não ter de estar a criar um acesso à base de dados em todos os DAO.

4.3 Modelo Lógico da Base de Dados



5 Vista

Como foi sugerido pelos professores nas aulas práticas, utilizámos as classes fornecidas "Menu"e "TextUI"para implementarmos uma vista para o nosso programa.



Para além de opções que implementam as funcionalidades requeridas para esta fase (opções 1 a 4), adicionámos outras opções para facilitar o povoamento da nossa base de dados (opções 5 a 9).

5.1 Menu

A classe Menu tem apenas como objetivo implementar métodos para "construir"o menu. Por exemplo, são definidos métodos para associar funções a opções do menu e métodos para definir pré-condições para as opções disponíveis.

5.2 TextUI

Esta classe tem como objetivo implementar a *front end* do nosso software. É nesta classe que definimos o que é apresentado no menu, as opções dadas ao utilizador, as mensagens de erro, ou seja, tudo o que for necessário para que aja boa interação entre o utilizador e o programa.

De forma a que o programa corra suavemente e sem erros por introdução de dados inválidos, foram criadas validações para o *input* como se pode ver em seguida, por exemplo quando o utilizador insere o código do corredor.

```
boolean flag = false;
do {
    if(flag) System.out.println("Corredor não existe, tente novamente!");
    System.out.println("Insira o código do Corredor: ");
    zona = scanner.nextLine();
    flag = true;
}while(!model.existeZona(zona));
flag = false;
```

6 Análise Crítica dos Resultados

Em geral, estamos contentes com a nossa implementação do programa. No entanto precisamos de realçar algumas dificuldades que tivemos em relação à coerência com a fase

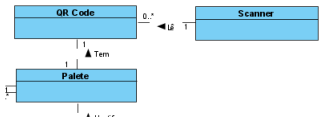
anterior e da conexão do nosso programa à base de dados.

Temos noção de que, na fase anterior, abordamos certos aspetos que são mais complicados do que o necessário. Como tínhamos que abordar todos os use cases, a nossa arquitetura visou fornecer a maior eficiência e organização possível aos mesmos. Ora, nesta fase, apesar de a equipa docente ter diminuído as funcionalidades do programa que tínhamos de implementar, de modo a termos coerência com os diagramas que realizámos na fase 2, nomeadamente os diagramas de classe e diagramas de sequência, tivemos de manter certos aspetos da nossa arquitetura, mesmo sendo desnecessários (na medida em que não demonstram grandes vantagens) para as funcionalidades que tivemos de implementar nesta fase. Um exemplo disto é o mapa do armazém. Na fase anterior tínhamos planeado utilizar um grafo para o representar, uma vez que possibilitava verificarmos qual o robô mais próximo da palete que necessitava de transporte e calcular os percursos ótimos utilizando teoria de grafos que foi estudada noutras unidades curriculares. Se tivéssemos optado por utilizar a sugestão dos professores nas aulas teríamos uma implementação mais simples, mas que não respeitaria a nossa arquitetura na fase anterior.

Outro aspeto da implementação que se revelou mais desafiante foi a conexão do nosso programa à base de dados. Houve vários motivos para isto: não só não temos muito experiência em trabalhar com bases de dados, mas também nunca utilizámos um conector para podermos utilizá-la no nosso programa. Apesar desta dificuldade, para além do que foi ensinado nas aulas, com algum estudo da documentação do conector conseguimos alcançar o nosso objetivo e, como resultado, a persistência de dados.

Agora que chegámos ao fim deste projeto, podemos realizar uma análise global do percurso efetuado nestas 3 fases. Revisitando o modelo de domínio podemos verificar que demos importância a mais a entidades menos relevantes para o nosso programa. Por exemplo, as entidades Camião e Camionista não foram utilizadas nas fases seguintes. Por outro lado, apesar de não termos colocado uma entidade prateleira, porque, inicialmente, íamos organizar as paletes unicamente por zonas de armazenamento, na fase 2, verificámos que era uma entidade essencial para o funcionamento do armazém. Quanto à análise de requisitos e diagramas da fase 2, e, como já foi referido anteriormente neste relatório, acreditámos terem sido bastante assertivos, o que se verificou por não ter sido necessário fazer alterações aos mesmos.

Outro modo de verificar esta coerência entre as várias fases deste projeto é focando-nos num use case do nosso programa. Deste modo, selecionando o use case de comunicar o código QR de uma palete que chegou ao armazém podemos observar o seguinte: No modelo de domínio da primeira fase podemos verificar que um scanner do armazém lê os QR codes que identificam uma dada palete. Este use case foi também explicitado no fluxo e diagrama de use case seguintes:



Modelo de Domínio

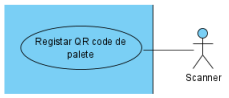
5.3.3 Registar QR code de palete

Use case: Registar QR code de palete
Pré condição: True
Pos condição: O sistema fica com um registo de uma nova palete
Descrição: O scanner lê e comunica ao sistema o código de uma palete descarregada

Fluxo Normal:
1 Scanner envia codigo da palete lido
2. O sistema regista uma nova palete

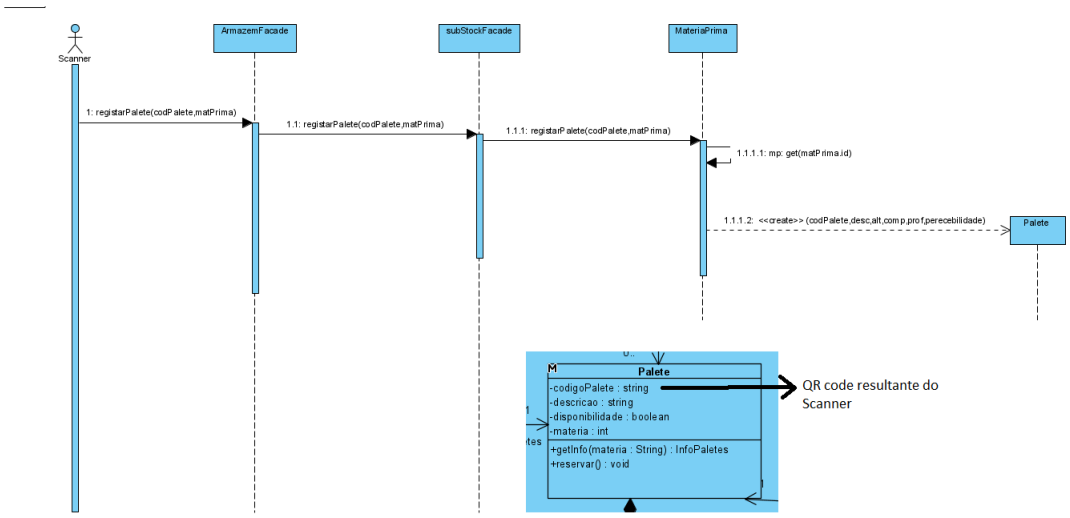
Fluxo de Exceção 1 [Leitura Invalida] (passo 1):
1.1 Scanner informa o erro ao sistema
1.2 Sistema regista o erro

Fluxo do use case



Modelo de use case

Ora, Podemos verificar que esta organização mantém-se na fase 2, como é vísivel no diagrama de classes resultante da análise de requisitos efetuada e no diagrama de sequência que representa este use case:



Na fase 3, ao implementar os DAO's, apercebemo-nos que o comportamento do método que implementa este use case iria ser ligeiramente diferente. Porém, na sua essência, é idêntico ao comportamento esperado e planeado nas fases anteriores.

Em jeito de conclusão, conseguimos implementar o programa pretendido e, mais importante, de acordo com o que tanto planeámos e arquitetámos nas fases anteriores. Com as classes fornecidas pelos docentes, facilmente criámos uma vista e, com o uso do conector do MySQL para o Java obtivémos a persistência dos dados do nosso programa.