Universidade do Minho
Mestrado Integrado em Engenharia Informática

DESENVOLVIMENTO DE SISTEMAS DE SOFTWARE
FASE 2
GRUPO 11

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

28 de novembro de 2020


A89586


A89585


A89572


A89564

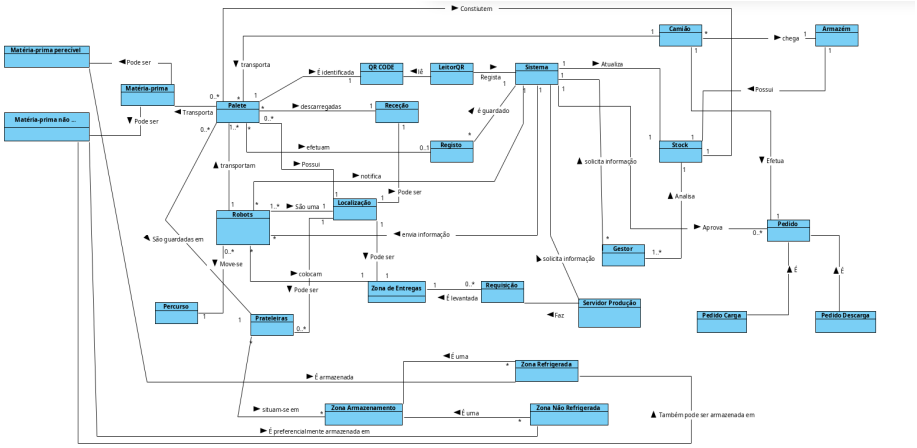
Conteúdo

1 Introdução

Neste relatório serão apresentados novos **Use Cases**, **Use Cases** corrigidos bem como **Diagramas de classes**, **Diagramas de componentes**, **Diagramas de sequência** e, por fim, **Diagrama de packages**, correspondentes aos **Use Cases** que serão implementados na nossa futura aplicação.
Os objetivos desta segunda fase do trabalho prático eram desenvolver uma arquitetura concetual do sistema, capaz de suportar os Use Case definidos. Além disso, também nos foi pedido para desenvolver a modelação comportamental dos métodos necessários à implementação dos nossos Use Cases.

2 Alteração ao Modelo de Domínio

Com a disponibilização de novos Use Case, que não havíamos definidos na primeira fase do nosso projeto, foi necessário realizar algumas pequenas alterações ao Modelo de Domínio, nomeadamente acrescentar as entidades LeitorQR e Servidor Produção. Em baixo, apresentamos o modelo de Domínio resutante destas alterações. Previamente, o Gestor fazia requisições e consulta a listagem de paletes, mas, neste novo modelo, ele apenas está responsável pela consulta de paletes, ficando a responsabilidade de fazer requisições a cargo do Servidor de Produção. O leitorQR foi acrescentado pois no nosso primeiro modelo não estava especificado que entidade é que realizava a leitura do QRCode e registava no sistema.



3 Use cases:

Apresentamos aqui as definições dos Use Case a usar nesta segunda fase, uma vez que tivemos que definir novos Use Case e redefinir Use Cases definidos na fase transata. Estas alterações em relação à primeira fase ajudaram a definir de uma melhor maneira os nossos Use Case e a definir Use Cases necessários que não haviam sido definidos previamente, como é o caso do início e término da sessão do Gestor no Sistema de Gestão do Armazém

3.1 Use case: Gestor inicia sessão

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

Fluxo alternativo 1: Gestor insere credenciais erradas (Passo 2)
1. Sistema informa que credenciais estão incorretas
2. Gestor insere de novo as credenciais
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

3.2 Use case: Gestor termina sessão

Cenário: Cenário 3
Pré-condição: O Gestor tem sessão iniciada
Pós-condção: Gestor termina sessão no sistema
Fluxo normal:
1. Gestor comunica ao sistema que quer terminar sessão.
2. Sistema pergunta ao gestor se quer confirmar o término da sessão
3. Gestor confirma término de sessão
4. Sistema termina sessão do gestor

Fluxo exceção: Gestor não confirma término de sessão (Passo 2)
1. Gestor não confirma o término da sessão
2. Sistema mantém sessão iniciada

3.3 Use case: Leitor comunica código QR

Cenário: Cenário 2
Pré-condição: True
Pós-condção: QR code é registado 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. Sistema gera um novo Código QR para a palete.
3. Palete é registada no sistema.

3.4 Use Case: Gestor solicita localização das paletes

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

3.5 Use case: Robot notifica recolha de palete

Descrição: Robot notifica sistema de que recolheu uma palete de uma prateleira.
Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete fica registada como estando a ser transportada
Fluxo normal:
1. Robot envia notifição ao sistema de recolha de palete
2. Sistema aprova a recolha da palete
3. Robot recolhe palete e transporta-a para o local suposto

Fluxo alternativo 1: Sistema não aprova a recolha da palete (passo 2)
2.1. Sistema informa robot de que a recolha não pode ser realizada.
2.2. Robot vai efetuar outras tarefas.
2.3. Sistema informa robot de que pode recolher a palete.
2.4. Robot recolhe palete e transporta-a para o local suposto

3.6 Use case: Robot notifica transporte de palete

Descrição: O robot notifica o sistema de que transporta uma palete da zona de receção até à prateleira predestinada ou da prateleira até à zona de carga.
Cenário: Cenário 2
Pré-condição: True
Pós-condição: Palete fica registada como estando a ser transportada e guardada na respetiva prateleira ou fica registada como entregue e guardadas na zona de carga
Fluxo normal:
1. Robot vê aprovada a recolha de palete
2. Robot transporta palte até à respetiva prateleira ou zona de carga
3. Robot armazena palete na prateleira ou zona de carga.
4. Robot notifica sistema do transporte com sucesso da palete.

Fluxo alternativo 1: Não existe prateleira disponível para armazenamento de palete (passo 2):
2.1. Robot coloca a palete na queue de paletes não armazenadas.
2.2. Robot vai efetuar outras tarefas.
2.3. Prateleira fica disponível.
2.4. Sistema notifica robot de que a prateleira já se encontra disponível.
2.5. Regresso ao passo 3.

3.7 Use case: Robot notifica entrega de paletes

Descrição: O robot notifica o sistema de que diexou, com sucesso, a palete que tinha recolhido e transportado.
Cenário: Cenário 2.
Pré-condição: True
Pós-condição: Palete é transportada para nova localização no armazém.
Fluxo normal:
1. Robot transporta a palete até ao novo destino
2. Robot deixa palete na nova localização
3. Robot notifica sistema da entrega da palete com sucesso.

Fluxo alternativo 1: Destino cheio (passo 2)
2.1. Robot aguarda com a palete até destino estar disponível

Fuxo exceção 1: Destino cheio (passo 2)
2.1 Robot transporta palete para uma localização que esteja disponível.
2.2. Robot notifica o sistema de que é necessário relocalizar palete.
2.3. Robot vai realizar outras tarefas.

3.8 Use case: Servidor de produção faz requisição de palete

Descrição: Servidor de produção envia pedido ao sistema de requisição de material.
Cenário: Cenário 2
Pré-condição: True
Pós-condição: Fica registado no sistema uma requisição de material para ser tratada.

Fluxo normal:
1. Cliente envia pedido de requisição de material.
2. Servidor de produção recebe o pedido.
3. Sistema processa o pedido verificando se existem os materiais requisitados.
4. Sistema coloca a requisição numa queue de requisições a enviar.

Fluxo alternativo 1: Não existem materiais requisitados (passo 3)
4.1. Sistema notifica servidor de produção de que não existem todos os materiais, mas que pode requisitar os materiais existentes.
4.2. Servidor de produção de que não existem todos os materiais, mas que pode requisitar os materiais existentes.
4.3. Cliente aceita nova requisição.
4.4. Nova requisição é processada.

Fluxo exceção 1: Cliente cancela requisição (passo 4.3)
4.3.1. Cliente cancela pedido.

4 Lógica de Negócio

- Pedir credenciais para efetuar o login.
- Validação das credendiciais de login do Gestor.
- Efetuar login no sistema.
- Terminar sessão no sistema.
- Validar QRCode.
- Registar palete no sistema.
- Gerar novo QRCode.
- Registar QRCodes.
- Listar paletes e sua localização.
- Notificar recolha de palete.
- Aprovar recolha de palete.
- Informar impossibilidade de recolha de palete.
- Notificar transporte bem sucedido.
- Notificar entrega bem sucedidida de palete.
- Efetuar pedido de requisição.
- Aprovar pedido de requisição.
- Notificar inexistência de materiais.
- Efetuar nova requisição.
- Cancelar requisição.

5 Métodos

+ consultaListagem(): List<Palete>
+ pedeLista(): List<Palete>
+ listaPalete(): List<Palete>
+ solicitaEntrega(codPalete: String): boolean
+ entregaPalete(codPalete: String): boolean
+ getPalete(codPalete: String) Palete
+ entregaPalete(): boolean
+ insereCredencial(email: String, password: String): Gestor
+ exibeCredenciais(): void
+ validaCredenciais(email: String, password: String): boolean
+ efetuaLogin(email: String, password: String): Gestor
+ terminaSessao(): booelan
+ pedeRecolha(codPalete: String): boolean
+ recolhePalete(codPalete: String): boolean
+ getPalete(codPalete: String): Palete
+ recolhePalete(): boolean
+ validaRecolha(): boolean
+ lePalete(codQR: QRCode): QRCode
+ validaQRCode(codQR: QRCode): boolean
+ geraQRCode(): QRCode

+ registaCodigos(cod: QRCode, codQR: QRCode): void
+ registaNovoQR(cod: QRCode, codQR: QRCode) void
+ registaPalete(cod: QRCode, palete: Palete): QRCode
+ registaPalete(palete: Palete): QRCode
+ registaCodigo(cod: QRCode): void
+ pedido(list: List<Material>): boolean
+ validaPedido(list: List<Material>): boolean
+ cancelaRequiscao(): void
+ pedeMaterialDisponivel(list: List<Material>): List<Material>
+ getDisponivel(list: List<Material>): List<Material>
+ fazRequisicao(list: List<Material>): Requisicao
+ registaRequisicao(list: List<Material>): Requisicao
+ aTransportar(codPalete: QRCode, zona: Zona): boolean
+ destinoDisponivel(codPalete: QRCode, zona: Zona): boolean
+ destinoLivre(zona: Zona): boolean

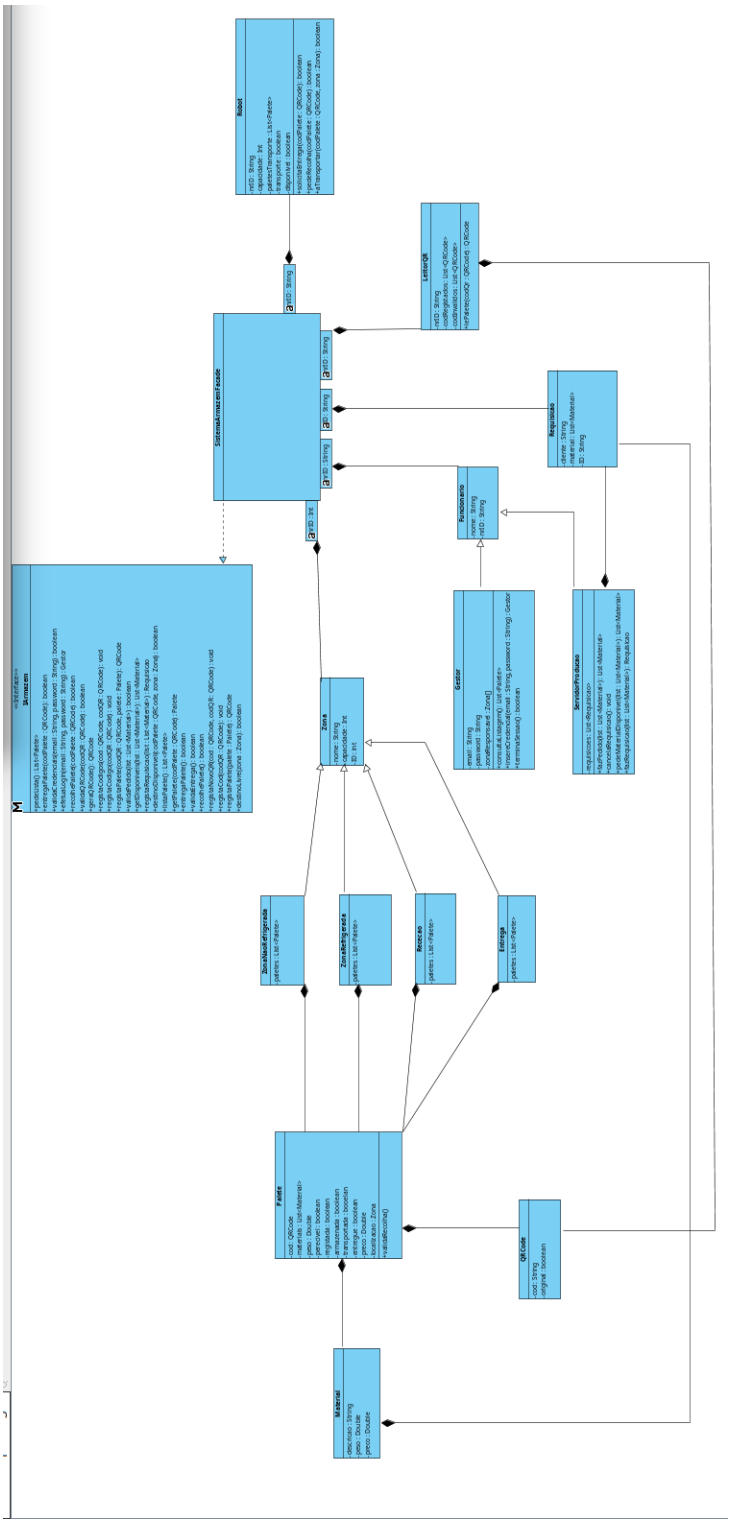
6 Diagrama de Classes

Definimos este diagrama com as classes representadas nas imagens abaixo. Os objetos LeitorQR, Robot, Funcionario, Zona e Requisicao são guardados sobre a forma de um Map, facilitando, assim , o seu acesso.
A superclasse zona divide-se em 4 subclasses(ZonaRefrigerada, ZonaNaoRefrigerada, Entrega e Rececao), tendo cada subclasse uma List das paletes que lá se encontram armazenadas. Cada zona é identificada por um ID e tem informação sobre a capacidade máxima que consegue suportar.
A classe Palete possui 10 atributos. O QRCode que a identifica é único para cada palete e esta possui também informações sobre o seu estado (se se encontra registada, armazenada, em transporte, ou entregue), sobre o preço total dos materiais que estão nela armazenados e sobre a sua localização.
O QRCode apenas possui dois atributos. O código em si, representado por uma String, e uma flag que indica se este QRCode é original ou se foi gerado pelo nosso sistema.
Referindo agora outra superclasse, Funcionario, divide-se em duas subclasses, Gestor e ServidorProducao.
O Gestor é um funcionário que necessita de efetuar login no nosso sistema e que pode consultar a listagem completa das paletes do armazém.
Já o servidor de produção está responsável por efetuar as requisições de material que chegam ao armazém. Temos, também, a classe LeitorQR que é identificada por um nrID do tipo String e que também tem como atributo os códigos QR que são dados como válidos e guarda, também, os códigos inválidos à medida que faz a leitura destes. Finalmente, temos a classe Robot, também identificada por um nrID do tipo String e que tem como atributos o número de paletes que consegue transportar, as paletes que está a transportar e se se encontra disponível para novas tarefas.

```java
public class Funcionario{
  private String nome;
  private String nrID;
}

public class Gestor extends Funcionario{
    private String email;
    private String password;
    private Zona[] zonaResponsavel;
}

public class ServidorProducao extends Funcionario{
  private List<Requisicao> requisicoes;
}

public class Zona{
  private String nome;
  private int capacidade;
  private int ID;
}

public class ZonaNaoRefrigerada extends Zona{
  private List<Palete> paletes;
}

public class ZonaRefrigerada extends Zona{
  private List<Palete> paletes;
}
```

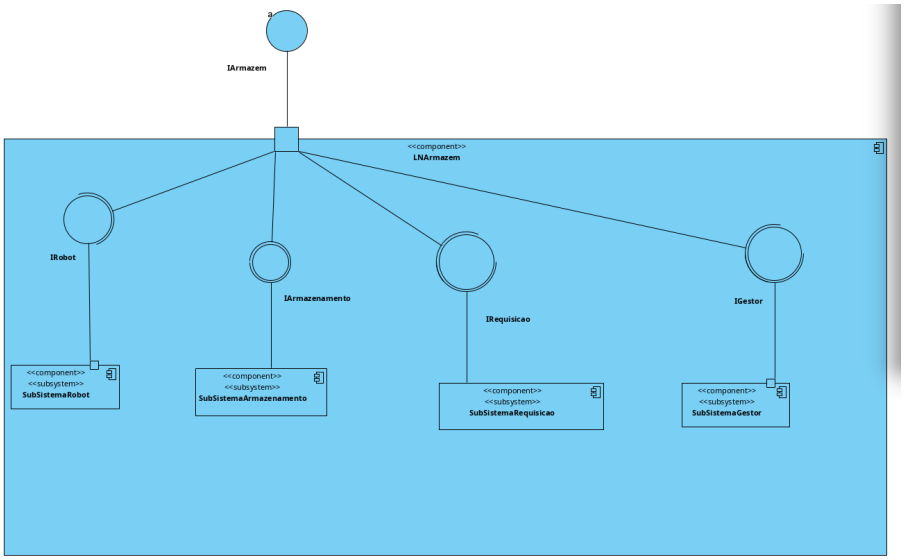
```java
public class Rececao extends Zona{
  private List<Palete> paletes;
}

public class Entrega extends Zona{
  private List<Palete> paletes;
}

public class Palete{
  private QRCode cod;
  private List<Material> materiais;
  private Zona localizacao;
  private double peso;
  private double preco;
  private boolean perecivel;
  private boolean registada;
  private boolean armazenada;
  private boolean transportada;
  private boolean entregue;
}

public class QRCode{
  private String cod;
}
```

```
55  public class Material{
56    private String descricao;
57    private double peso;
58    private double preco;
59  }
60
61  public class Requisicao{
62    private String cliente;
63    private List<Material> material;
64    private String ID;
65  }
66
67  public class LeitorQR{
68    private String nrID;
69    private List<QRCode> codRegistados;
70    private List<QRCode> codInvalidos;
71  }
72
73  public class Queue{
74    private String descricao;
75    private List<Palete> paletes;
76  }
77
78  public class Robot{
79    private String nrID;
80    private int capacidade;
81    private List<Palete> paleteTransporte;
82    private boolean transporta;
83    private boolean disponivel;
```



7 Diagrama de Componentes

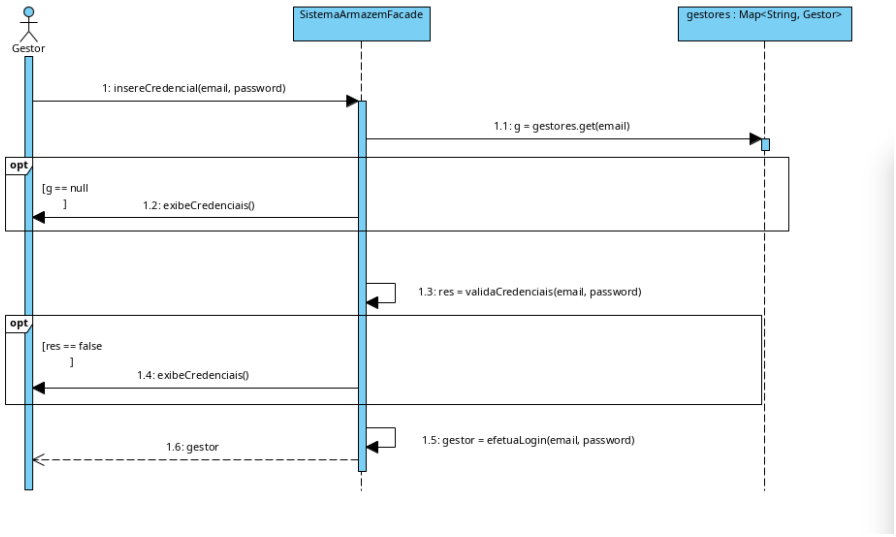
Neste diagrama, identificamos 4 componentes distintos. Definimos os componentes SubSistemaRobot, SubSistemaGestor, SubSistemaArmazenamento e SubSistemaRequisicao. Decidimos a criação destes 4 componentes para permitir dividir o nosso programa em 4 partes distintas. Uma parte mais ligada à consulta do stock disponível, representada pelo SubSistemaGestor, outra parte ligada ao transporte, recolha e entrega de paletes, representada pelo SubSistemaRobot, mais uma parte ligada ao registo e armazenamento das paletes, ligada a SubSistemArmazenamneto e, finalmente, uma parte mais ligada às requisições de material que são efetuadas, associada ao SubSistemaRequisicao.
O SubSistemaRobot está responsável pelas ações ligadas ao robot e ao transporte, recolha e entrega de paletes. Tendo em si os métodos necessários para realizar estas ações, achamos pretinente que este componente fosse criado.
No SubSistemaGestor poderemos identificar o ator Gestor e este componente está responśavel pela listagem de palete e todos os processos que envolve a gestão do stock do armaźem. Tem em si, também, os processos ligados ao login e logout do Gestor.
No SubSistemaArmazenamento, será possível identificar o ator LeitorQR, uma vez que este é responśavel pelo registo de novas palete. Este componente é responsável por todos os processos ligados ao armazenamento e novos registos de paletes.
Finalmente, o SubSistemaRequisicao tem em si o ator Servidor de Produção e é responsável por todos os processos ligados à requisição de materiais.



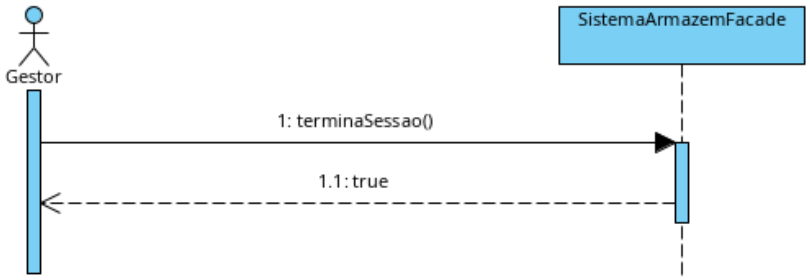
8 Diagramas de Sequência

Também definidos 8 diagramas de sequência para, deste modo, termos um diagrama de sequência para cada Use Case apresentado. Com diferentes níveis de complexidade, estes diagramas permitiram identificar as interações entre os diferentes objetos, bem como as mensagens trocadas entre eles. Com a definição destes diagramas, foi possível analisar a dsitribuição das responsabiliades e tarefas de cada objeto, permitindo, assim, ter uma melhor noção de como todos os processos estão a ser efetuados, temporalmente, conseguindo, também, analisar cada interação existente no programa, bem como as mensagens enviadas e recebidas por cada objeto.

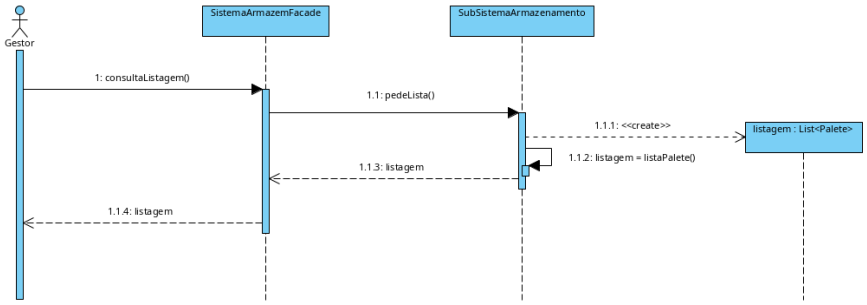
8.1 Iniciar Sessão



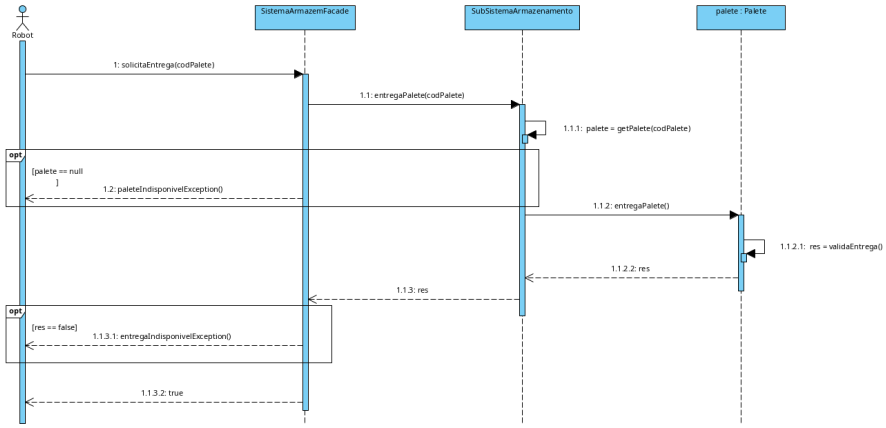
8.2 Terminar Sessão



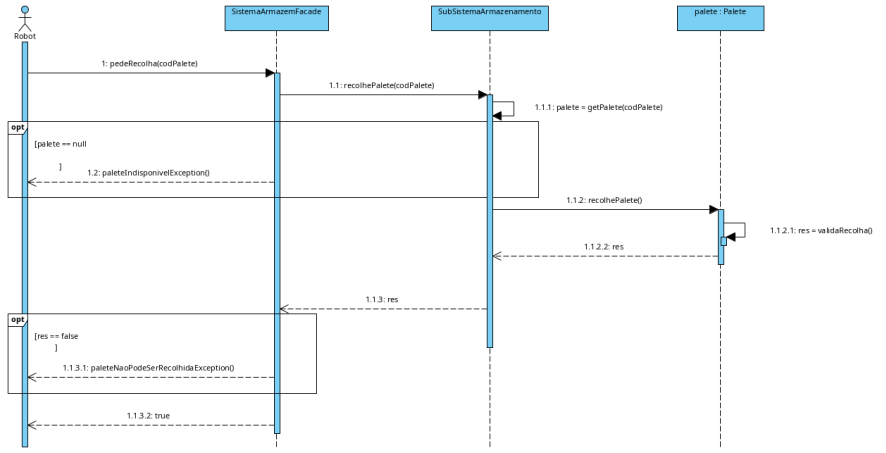
8.3 Consultar Listagem



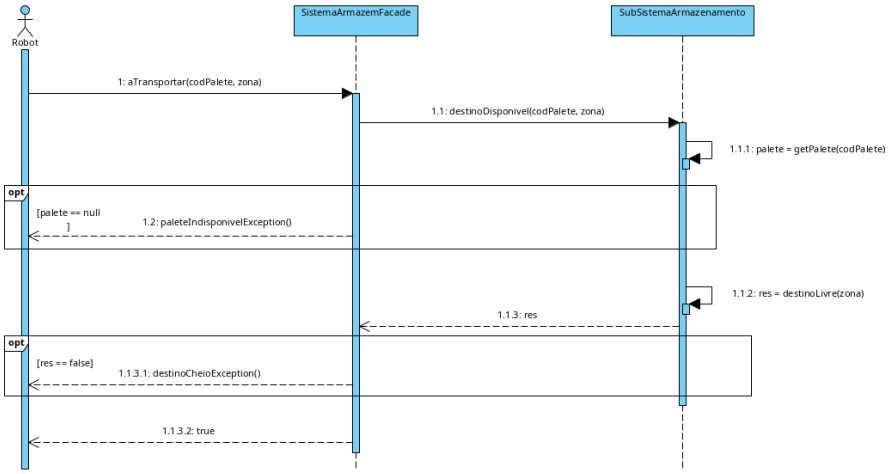
8.4 Notifica Entrega de Palete



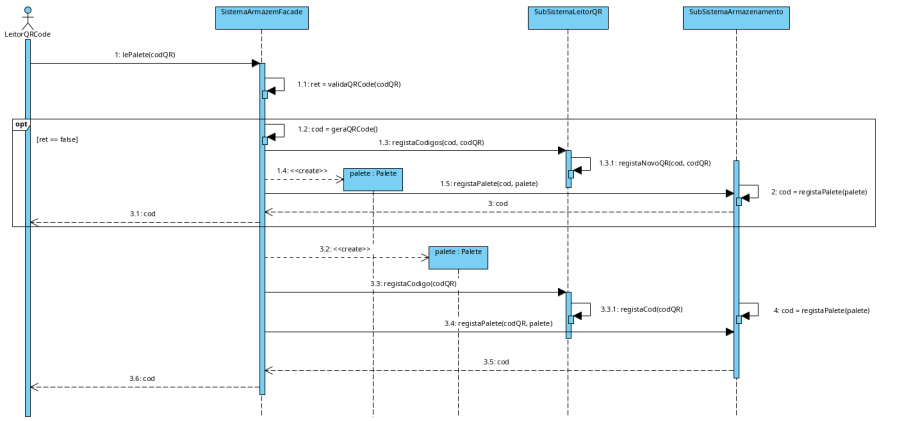
8.5 Notifica Recolha Palete



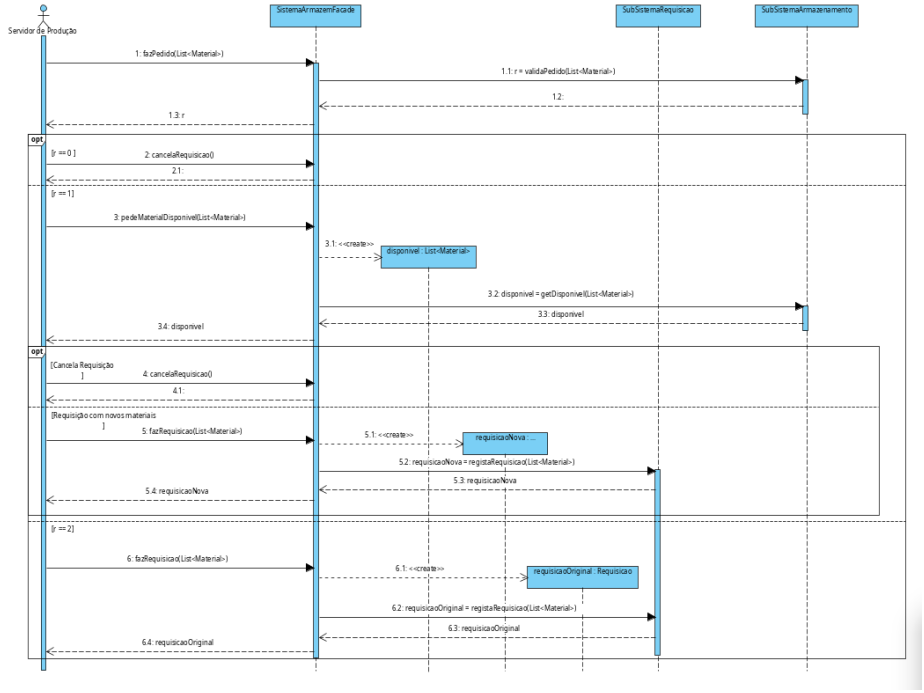
8.6 Notifica Transporte Palete



8.7 Regista Palete

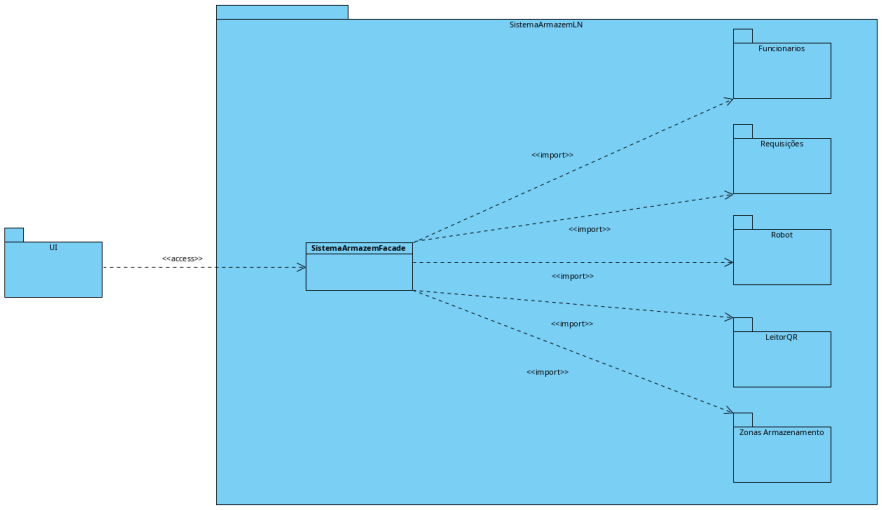


8.8 Efetua Requisição



9 Diagrama de Packages

Apresentamos, agora, o diagrama de packages, onde é possível observar e analisar a divisão lógica do programa. Podemos observar que existem 2 grandes packages, sendo eles a UI e o SistemaArmazemLN.



9.1 Package UI

Neste package encontra-se a implementação gráfica que irá permitir a interação entre o utilizador e o sistema.

9.2 SistemaArmazemLN

Neste package encontra-se a lógica de negócio do nosso sistema de gestão do armazém, responsável por todas as ações do sistema. Dentro deste package foi necessário definir mais 6 packages.

9.2.1 SistemaArmazemFacade

Responsável pela ligação entre a UI e a parte lógica do nosso sistema.

9.2.2 Funcionários

Tratamento e armazenamento de todos os dados referentes a funcionários, nomeadamente, gestores e servidores de produção.

9.2.3 Robot

Tratamento e armazenamento de todos os dados referentes aos robots

9.2.4 LeitorQR

Tratamento e armazenamento de todos os dados referentes ao leitorQR

9.2.5 Requisições

Tratamento e armazenamento de todos os dados referentes às requisições efetuadas.

9.2.6 Zonas Armazenamento

Tratamento e armazenamento de todos os dados referentes às zonas onde estão guardadas as paletes.

10 Conclusão e análise crítica dos resultados obtidos

Concluída esta segunda fase do trabalho prático, apesar de termos sentido mais dificuldades e termos considerado esta fase mais exigente do que a fase transata, percebemos a importância de uma boa estruturação de uma aplicação. Os diferentes diagramas desenvolvidos e todas as análises feitas nesta segunda fase permitiram, de algum modo, construir o esqueleto da nossa aplicação e perceber de que forma os diferentes objetos e entidades iram agir entre si. Permitiu, também, idealizar a estrutura do nosso trabalho através da definição dos diferentes subsistemas, sabendo assim as responsabilidades de cada um e de que forma a nossa aplicação irá ficar organizada.

Apesar dos resultados não terem sido tão bons como desejavamos, consideramos que conseguimos definir com sucesso os objetivos desta segunda fase e cremos que será muito importante para na útlima fase implementarmos a nossa aplicação, pois a estrutura e modelação já estão todas bem definidas, facilitando, assim, a implementação e desenvolvimento de toda a nossa aplicação.