Universidade do Minho

Departamento de Informática

Trabalho Prático

Desenvolvimento de Sistemas de Software

Joel Martins
a89575

Leonardo Marreiros
a89537

Manuel Moreira
a89471

Sara Dias
a89544

Conteúdo

Introdução 4

Modelo de Domínio 5

Modelo de Use Cases 6

Especificações dos Use Cases 8

Notificar recolha de palete 8

Notificar entrega de palete 8

Sistema notifica transporte de palete 9

Ler código de palete 10

Autorizar descarga 10

Solicitar listagem de paletes 10

Solicitar autorização de recolha 11

Solicitar autorização de descarga 11

Autorizar recolha 11

Notificar satisfação de requisição 12

Iniciar Sessão 12

Terminar Sessão 13

Registar 13

Diagrama de Componentes 14

Diagramas de Sequência 14

1. **Atualiza Localização Palete** 14
2. **Calcula percurso** 14
3. **Cancela palete** 15
4. **Cancela pedido** 15
5. **Conclui transporte** 15
6. **Comunica percurso** 16
7. **Consulta paletes** 16
8. **Efetuar requisição de material** 16
9. **Verificar disponibilidade para descarga** 17
10. **Verifica disponibilidade para recolha** 17
11. **Iniciar sessão** 17
12. **Procurar prateleira disponível** 18
13. **Procura robot** 18
14. **Regista palete** 18

15. **Regista paletes de pedido** 19
16. **Regista funcionário** 19
17. **Terminar sessão** 19
18. **Transporta palete** 19
19. **Trata pedido** 20
20. **Valida disponibilidade de matérias requisitadas** 20

Diagrama de Classes 21

Diagrama de ORM 22

Diagrama de Packages 23

Diagrama de Sequência de Implementação 24

1. **concluiTransportePalete** 25
2. **consultaPrateleiras** 25
3. **existePrateleira** 26
4. **reservaPorCodigo** 26
5. **existemPaletesATransportar** 27
6. **iniciaTransportePalete** 27
7. **leitorRegisto** 28
8. **calculaPercurso** 28
9. **calculaPercursoArmazenamento** 29
10. **calculaPercursoRecolha** 29
11. **procuraPrateleira** 30
12. **reservaPrateleiraPorCodigo** 30
13. **updatePrateleiras** 31
14. **movimenta** 31
15. **transportaPalete** 32
16. **verificaDisponibilidadeRobot** 33
17. **verificaExistenciaPalete** 33

Base de Dados 34

Implementação 35

Análise Crítica 38

Conclusão 39

Introdução

Este relatório foi desenvolvido no âmbito da realização do trabalho prático da unidade curricular de Desenvolvimento de Sistemas de Software. Sendo esta a última fase do projeto, e uma vez que foram feitas várias alterações nos diagramas apresentados nas fases anteriores à medida que foram encontrados diferentes problemas, decidimos voltar a apresentar estes mesmos diagramas, primeiro na versão apresentada na fase anterior, seguida da versão final e explicação das mudanças efetuadas. O projeto consistiu na modelação e implementação de um sistema de gestão de stocks de um armazém de uma fábrica. Para a modelação utilizamos a UML, Linguagem de Modelagem Unificada, para elaborar os diagramas que fazem parte este projeto. Além disso utilizamos o MySQL Workbench para o modelo lógico da base de dados. Esta aplicação foi desenvolvida em Java e a sua base de dados inerente em SQL.

Com este relatório pretendemos demonstrar a forma como abordamos a modelação do sistema apresentado. Adicionalmente, apresentamos também os modelos UML resultantes, assim como, as principais mudanças ao longo das diferentes iterações.

Objetivamente, na terceira e última fase, foram atualizados e implementados novos diagramas e foi feita a implementação de parte das funcionalidades pretendidas para o sistema, em concreto, foi implementada a parte referente ao armazenamento de paletes e consulta de locais ocupados.

Ao longo do relatório vamos voltar a apresentar os modelos de domínio, Use Cases, diagramas de sequência, diagrama de componentes e finalmente o diagrama de packages, juntamente com os novos diagramas, assim como as decisões tomadas e problemas encontrados.

Modelo de Domínio

Tal como dito anteriormente, o problema em questão consiste em desenvolver um sistema de gestão de stocks de um armazém de uma fábrica. Após analisar o enunciado disponibilizado, concluímos que era necessário a existência de vários conceitos a incluir na modelação do problema: Palete, Robot, Localização, Gestor, Prateleira (refrigerada ou não), Encarregado, Motorista, Servidor de produção, Matéria-prima (perecível ou não), Percurso, etc. Com isto, foram definidas as relações entre os conceitos referidos em cima:

- Uma palete é transportada por um robot. Da mesma forma, um robot só transporta uma palete de cada vez.
- Uma palete é caracterizada por uma altura, uma matéria-prima e um código.
- Uma palete está numa Localização.
- Uma Localização pode ser Zona de receção, Zona de entrega ou um corredor com armazenamento com prateleiras refrigeradas ou prateleiras não-refrigeradas.
- Cada prateleira tem uma altura.
- O Servidor de produção faz pedidos de uma ou mais matérias-primas.
- O Gestor autoriza uma descarga.
- O motorista descarrega uma ou mais paletes.
- Etc…

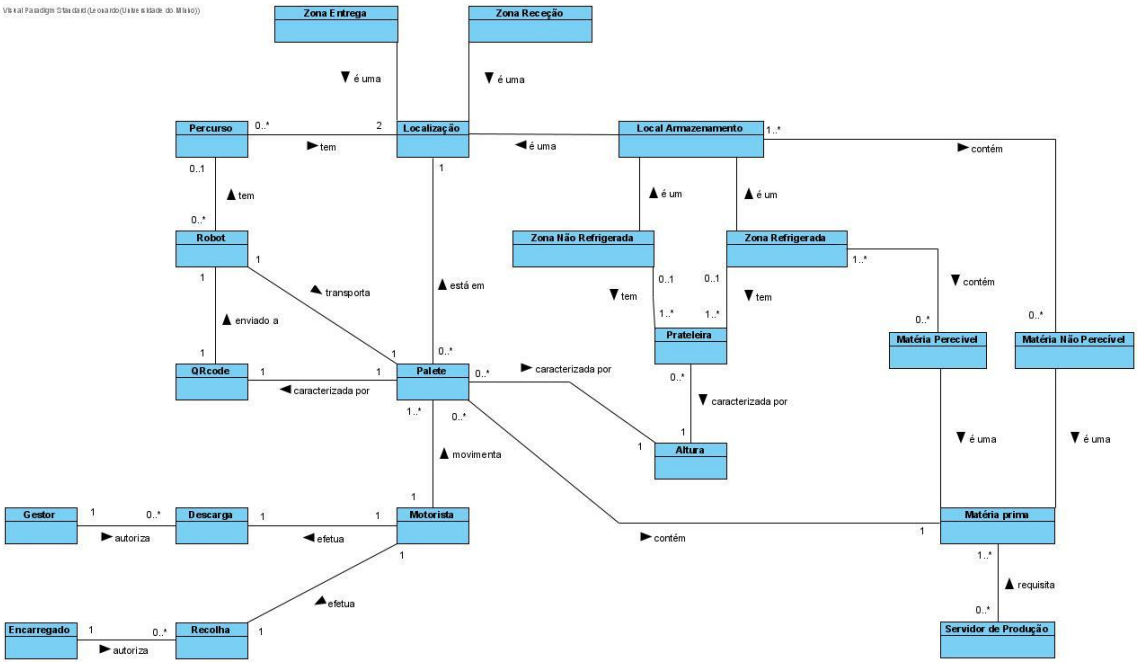


Figura 1 - Modelo de Domínio

Modelo de Use Cases

O diagrama de Use Cases representa os atores do Sistema e as tarefas/ações que cada um pode realizar. Consideramos os seguintes atores, associados a uma breve descrição:

- **Robot:** faz o transporte de paletes entre diversas localizações (zona de entregas, zona de receção e prateleiras, que nomeadamente se encontram em zonas refrigeradas ou não refrigeradas), notificando o Sistema a cada ação que executa.
- **Leitor de QRcodes:** lê o QRcode de uma palete para a registar no Sistema e posteriormente este poder identificá-la.
- **Encarregado:** autoriza as recolhas de matéria no armazém e notifica o Sistema das saídas da mesma.
- **Gestor:** responsável pela zona de receção do armazém, onde autoriza as descargas de matéria com base na listagem (para ter acesso à ocupação) e avisa o Sistema que existem paletes recém-chegadas para serem armazenadas.
- **Motorista:** transporta materiais para descarga no armazém ou recolhe materiais requisitados do armazém.
- **Servidor da Produção:** responsável por fazer as requisições de materiais ao armazém.

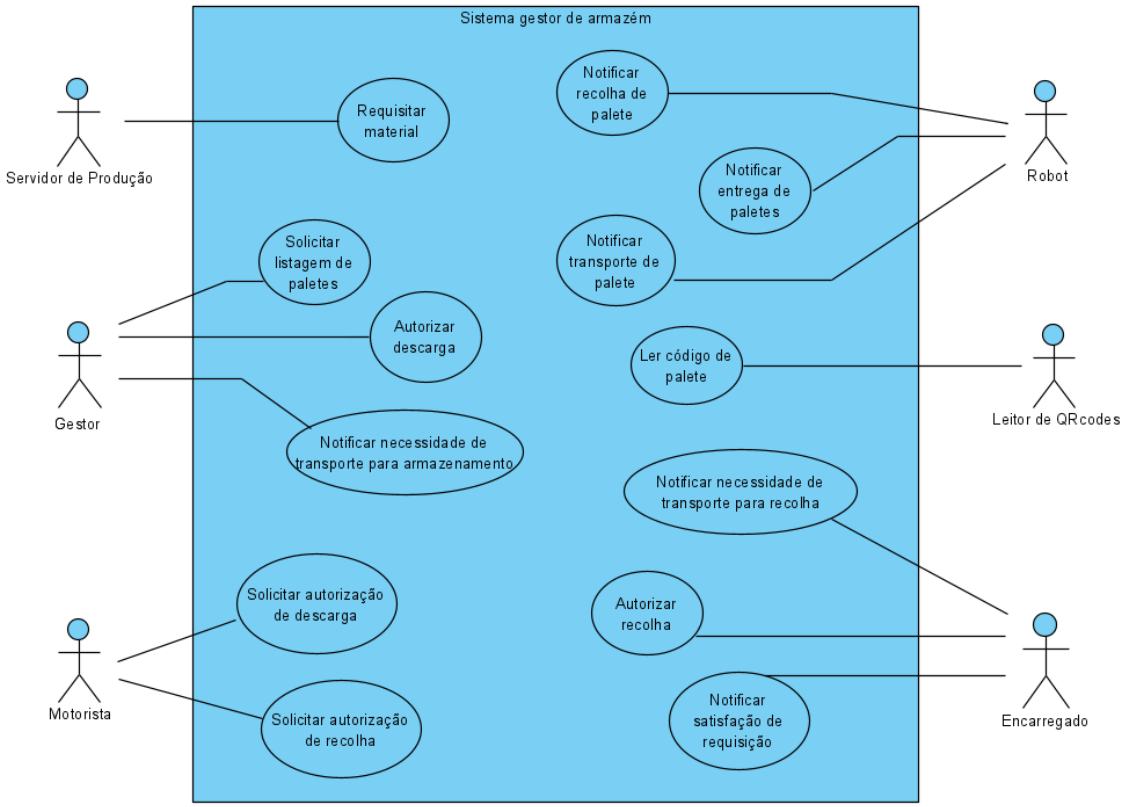


Figura 2 - Modelo de Use Cases

Desde a última fase, com o surgimento de novos problemas ou situações que não foram consideradas, foram adicionados atores, atualizados, adicionados e removidos Use Cases. Quanto aos Uses Cases, foi removido o Use case “Notificar necessidade de transporte para armazenamento” uma vez que o consideramos ser um pouco redundante: o Gestor está a informar que existem paletes à espera de transporte quando na realidade essa informação já faz parte do Sistema. Da mesma forma, foi removido o Use Case “Notificar necessidade de transporte para recolha” com a mesma justificação. Além disso foram adicionados os Use Cases para Iniciar Sessão, Terminar Sessão e Registar. Uma vez que estes Use Cases estariam presentes não só no Gestor como no funcionário, decidimos introduzir um novo Ator que funciona como generalização das entidades referidas anteriormente, desta forma prevenimos a duplicação de código. Alguns dos Use Cases foram atualizados de forma a conseguir tratar da forma pretendida cada caso. Nos casos em que isso acontece é apresentada uma breve explicação das mudanças efetuadas nas especificações apresentadas a seguir.

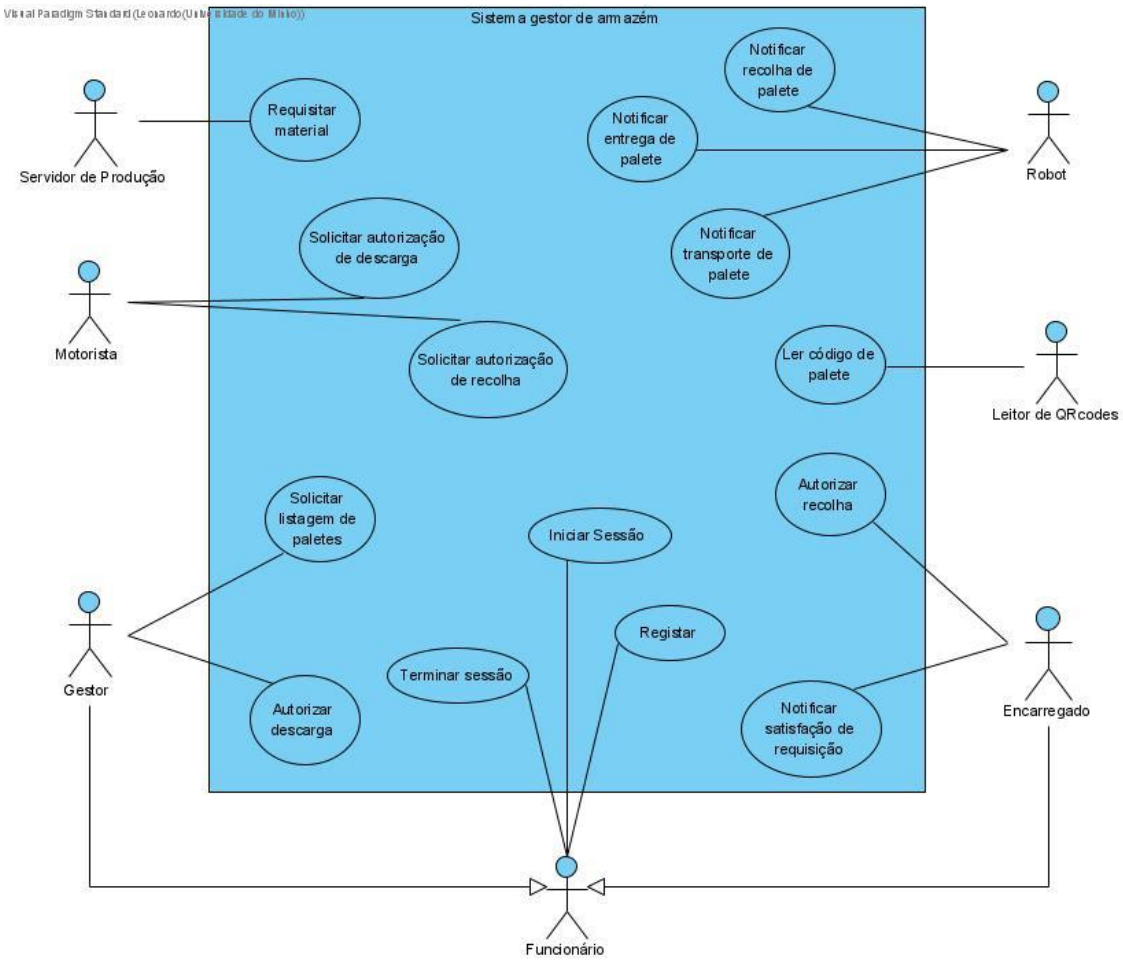


Figura 3 - Modelo de Use case versão final

Especificações dos Use Cases

Notificar recolha de palete

Descrição: O Robot notifica que recolheu a palete.

Cenários: É enviado um código de uma palete ao robot para este recolher.

Pré-Condição: A palete encontra-se no armazém.

Pós-Condição: A palete está a ser transportada.

Fluxo normal:

1. O Robot carrega a palete.

2. O Robot notifica o Sistema que recolheu a palete com sucesso.

Fluxo exceção 1 [armazém cheio] (passo 2):

1. Indica que o armazém está lotado pelo que o robot deixa de carregar a palete.

Fluxo exceção 2 [queue de paletes vazia] (passo 2):

1. Queue de paletes à espera de transporte encontra-se vazia pelo que não há paletes para entregar.

Fluxo exceção 3 [código de palete inexistente] (passo 2):

1. Código de palete enviado ao robot para recolha não consta na queue de paletes à espera de transporte.

Neste Use Case foram acrescentados os fluxos de exceção de forma a conseguir tratar melhor estes casos especiais que nos percebemos que poderiam acontecer.

Notificar entrega de palete

Descrição: O Robot notifica que foi entregue a palete no destino.

Cenários: O Robot transporta a palete para o destino.

Pré-Condição: O Robot recolheu a palete.

Pós-Condição: O Robot entregou a palete no destino.

Fluxo normal:

1. O Robot carrega a palete até ao destino.

2. O Robot notifica Sistema de entrega realizada com sucesso.

Fluxo exceção 1 [Não tem palete a entregar] (passo 2):

2. O robot não recolheu qualquer palete nem está a carregar nenhuma palete. Nada é entregue

Fluxo exceção 3 [código de palete inexistente] (passo 2):

2. Código de palete enviado ao robot para transporte não consta na queue de paletes à espera de transporte.

Neste Use Case foram acrescentados os fluxos de exceção de forma a conseguir tratar melhor estes casos especiais que nos percebemos que poderiam acontecer.

Sistema notifica transporte de palete

Descrição: O Sistema notifica o Robot da necessidade de transporte de palete.

Cenários: Queue de paletes à espera de transporte não está vazia.

Pré-Condição: A palete está no armazém.

Pós-Condição: O Robot recebe a notificação de transporte.

Fluxo normal:

1. O Sistema procura Robot para transportar palete.

2. O Sistema calcula percurso para o Robot.

3. O Sistema retira a palete da fila de paletes a aguardar transporte.

4. O Sistema comunica ao Robot a palete a transportar, o seu local de destino e o percurso a fazer.

Fluxo alternativo 1: [nenhum Robot disponível] (passo 3)

3.1. O Sistema aguarda disponibilidade de algum Robot.

3.2. Regressar a **4**.

Ler código de palete

Descrição: O Leitor de QRcodes lê o código da palete.

Cenários: Palete chega ao armazém.

Pré-Condição: True

Pós-Condição: Sistema fica com o registo do código da palete.

Fluxo normal:

1. O Leitor de QRcodes lê o código da palete.

2. O Leitor de QRcodes regista no Sistema a palete.

Autorizar descarga

Descrição: O Gestor processa o pedido de descarga de matéria prima no armazém.

Cenários: O Gestor autoriza descarga para o armazém.

Pré-Condição: Ocorreu solicitação de descarga.

Pós-Condição: Material descarregado.

Fluxo normal:

1. O Sistema notifica o Gestor de pedido de descarga.

2. O Gestor verifica ocupação das prateleiras.

3. O Gestor autoriza a descarga.

4. O Sistema adiciona as paletes à fila de paletes a aguardar transporte.

Fluxo alternativo 1: [nenhum Gestor disponível] (passo 1)

1.1. O Sistema notifica Motorista de indisponibilidade dos Gestores.

1.2. O Sistema aguarda disponibilidade de algum Gestor.

1.3. Regressar a 2.

Fluxo de exceção 1: [prateleiras totalmente ocupadas] (passo 3)

3.1. O Gestor recusa a descarga.

Solicitar listagem de paletes

Descrição: O Sistema processa listagem de paletes com a sua localização no armazém.

Cenários: O Gestor pede a listagem das paletes existentes no armazém.

Pré-Condição: True.

Pós-Condição: O Gestor obtém listagem com a localização das paletes do armazém.

Fluxo normal:

1. O Sistema consulta a localização de cada palete existente no armazém.
2. O Sistema envia listagem ao Gestor.

Solicitar autorização de recolha

Descrição: O Motorista pede autorização para recolher matéria prima.

Cenários: O Motorista chega ao armazém para recolher paletes.

Pré-Condição: True.

Pós-Condição: Sistema recebe notificação da autorização para recolha de paletes.

Fluxo normal:

1. O Motorista informa ao Sistema que está pronto para a recolha do material requisitado.

Solicitar autorização de descarga

Descrição: O Motorista pede autorização para descarregar matéria prima.

Cenários: O Motorista chega ao armazém para efetuar descarga.

Pré-Condição: True

Pós-Condição: Sistema recebe notificação da autorização para descarregar paletes.

Fluxo normal:

1. O Motorista informa ao Sistema que está pronto para descarregar material.

Autorizar recolha

Descrição: O Encarregado responde ao pedido de autorização de recolha.

Cenários: O Encarregado autoriza o levantamento do material requisitado.

Pré-Condição: Material requisitado encontra-se na zona de entrega.

Pós-Condição: Encarregado autoriza recolha.

Fluxo normal:

1. O Sistema notifica o Encarregado que vai ser efetuada uma recolha.

2. O Encarregado autoriza a recolha.

Fluxo alternativo 1: [nenhum Encarregado disponível] (passo 1)

1.1. O Sistema notifica Motorista de indisponibilidade dos Encarregados.

1.2. O Sistema aguarda disponibilidade de algum Encarregado.

1.3. Regressar a **2**.

Notificar satisfação de requisição

Descrição: O Encarregado notifica o Sistema sobre a saída das matérias primas do armazém.

Cenários: O Encarregado avisa o Sistema de que as paletes foram entregues.

Pré-Condição: Paletes foram entregues.

Pós-Condição: Sistema recebe notificação da entrega das paletes.

Fluxo normal:

1. O Encarregado notifica o Sistema da recolha das paletes.

Iniciar Sessão

Descrição: Permite o Funcionário autenticar-se no Sistema.

Cenários: O Funcionário pretende aceder à sua conta do Sistema.

Pré-Condição: True.

Pós-Condição: O Funcionário foi autenticado com sucesso.

Fluxo normal:

1. O Funcionário insere as suas credenciais.

2. O Sistema valida as credenciais.

3. O Sistema informa que entrou no Sistema.

Fluxo Exceção [Credenciais erradas] (Passo 2):

1. Mensagem de erro.

Terminar Sessão

Descrição: Permite o Funcionário sair do Sistema.

Cenários: O Funcionário pretende sair da sua conta do Sistema.

Pré-Condição: O Funcionário está autenticado no Sistema.

Pós-Condição: O Funcionário foi desconetado do Sistema com sucesso.

Fluxo normal:

1. O Funcionário pede para desconetar.
2. O Sistema desconeta o Funcionário.
3. O Sistema informa que foi feito o logout.

Registar

Descrição: Registar Funcionário.

Cenários: Novo Funcionário pretende criar uma conta.

Pré-Condição: True.

Pós-Condição: A conta foi criada com sucesso.

Fluxo normal:

1. O Funcionário insere o email e password.
2. O Sistema mostra uma mensagem de confirmação.
3. O Funcionário responde que sim.
4. O Sistema informa que o Funcionário foi registado com sucesso.

Fluxo Exceção [Credenciais erradas] (Passo 3):

2. O Sistema não regista o Funcionário.

Diagrama de Componentes

Para os vários métodos de use cases considerados na lógica de negócio, conseguimos identificar quatro subsistemas: Funcionários, Palete, Robots e Pedidos. A partir daí, construímos o seguinte diagrama de componentes:

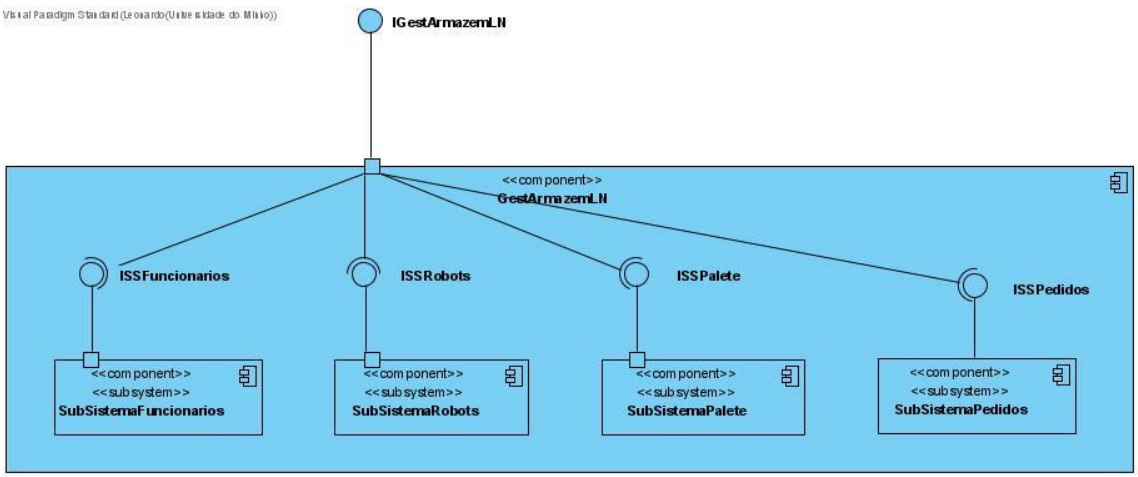
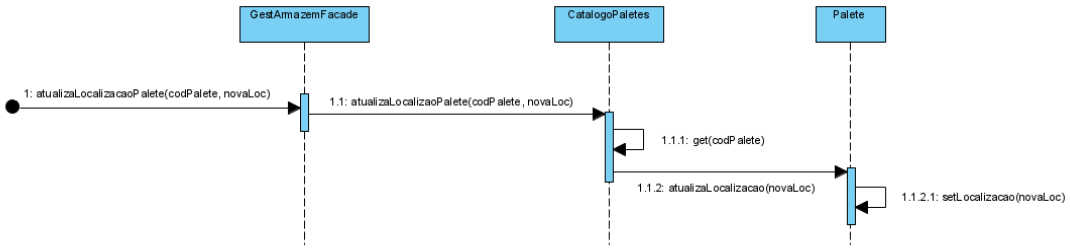


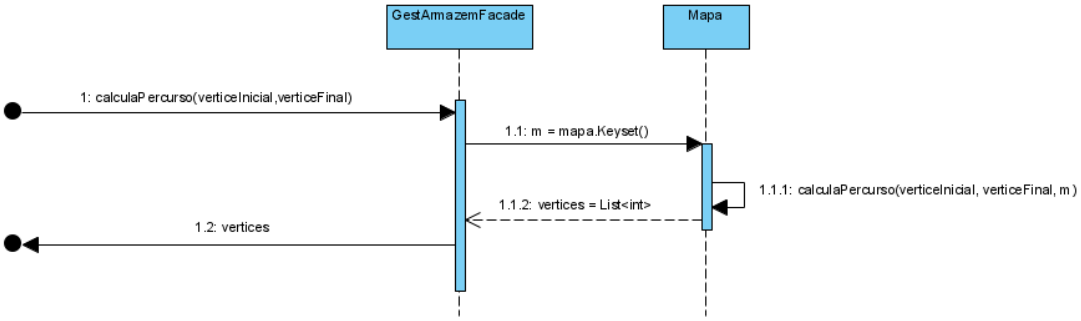
Figura 4 - Diagrama de Componentes

Diagramas de Sequência

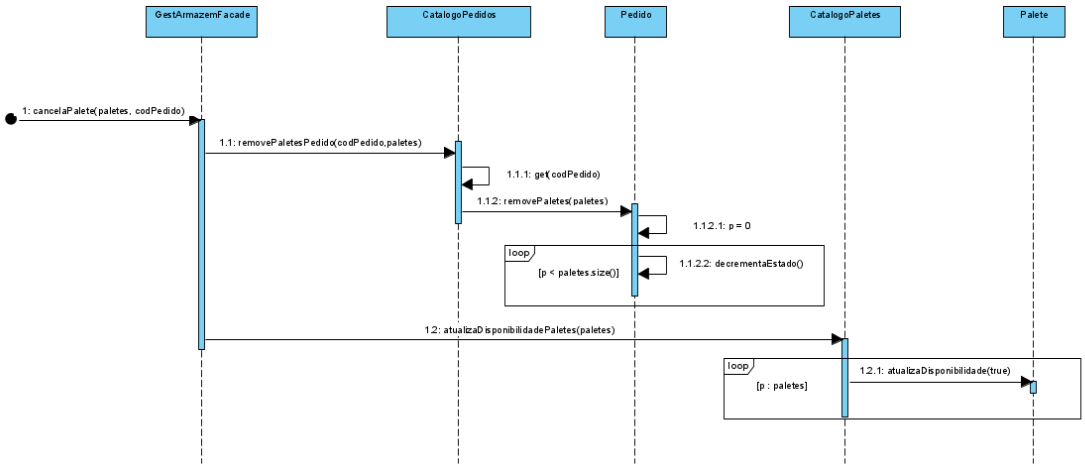
1. Atualiza Localização Palete



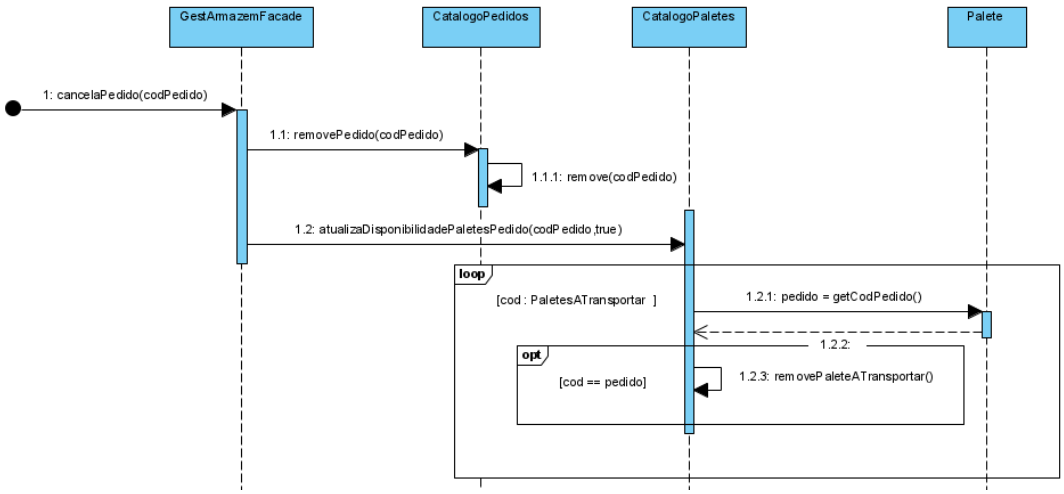
2. Calcula percurso



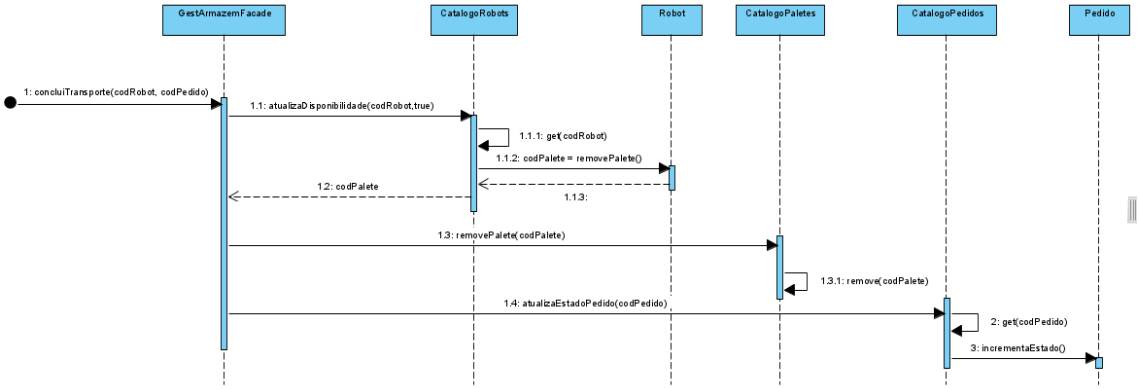
3. Cancela palete



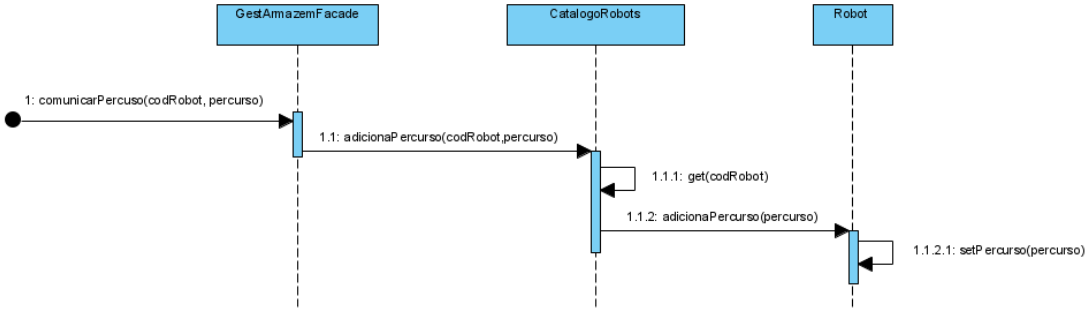
4. Cancela pedido



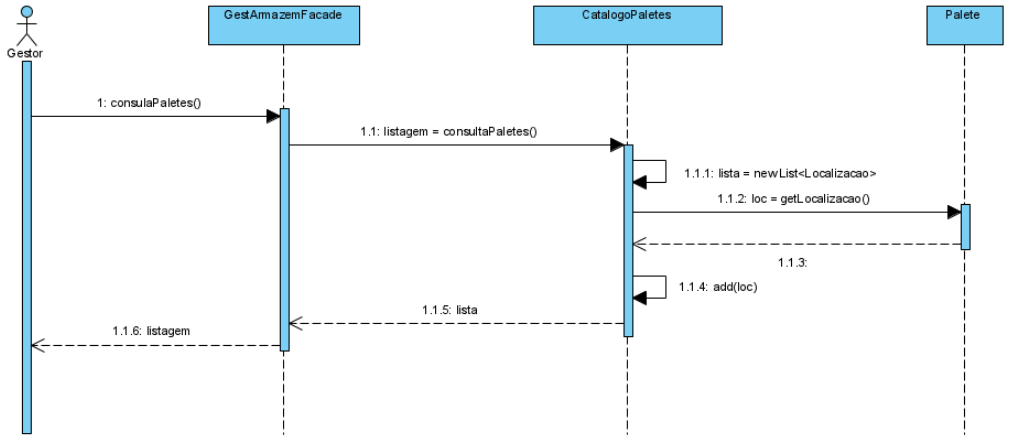
5. Conclui transporte



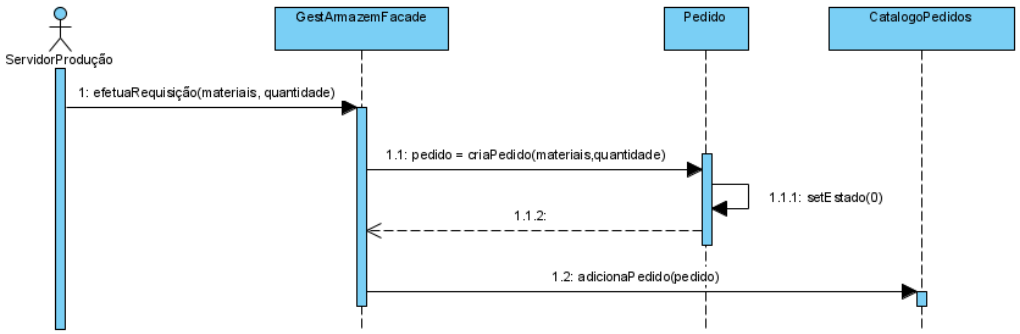
6. Comunica percurso



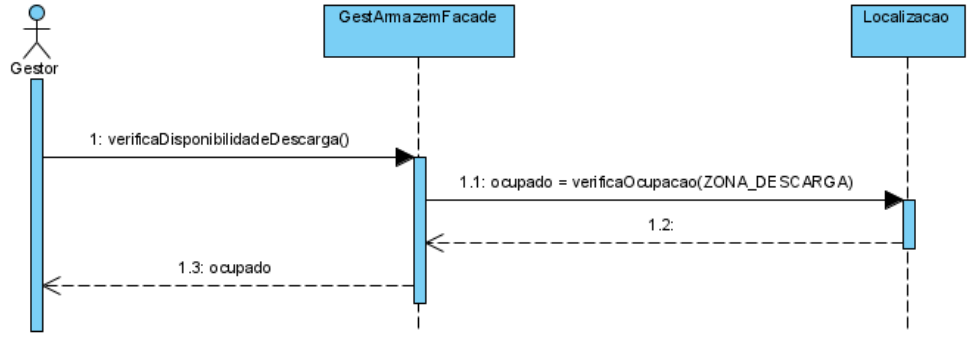
7. Consulta paletes



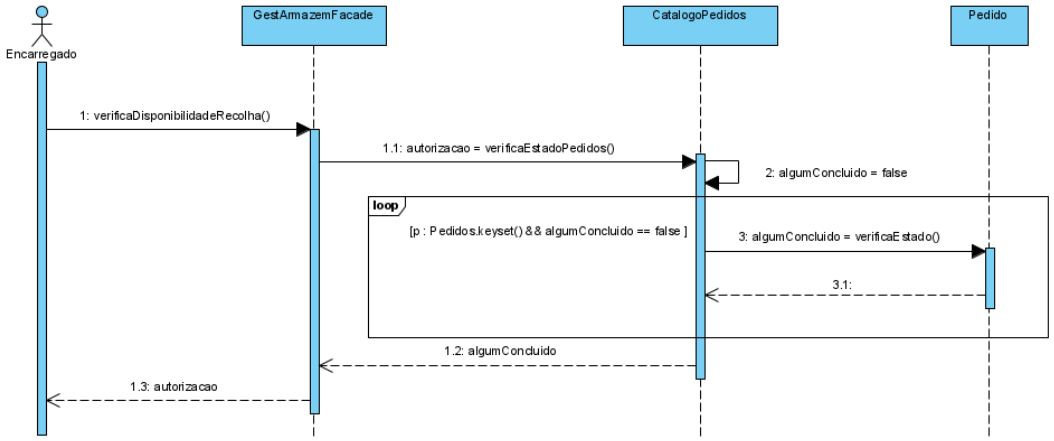
8. Efetuar requisição de material



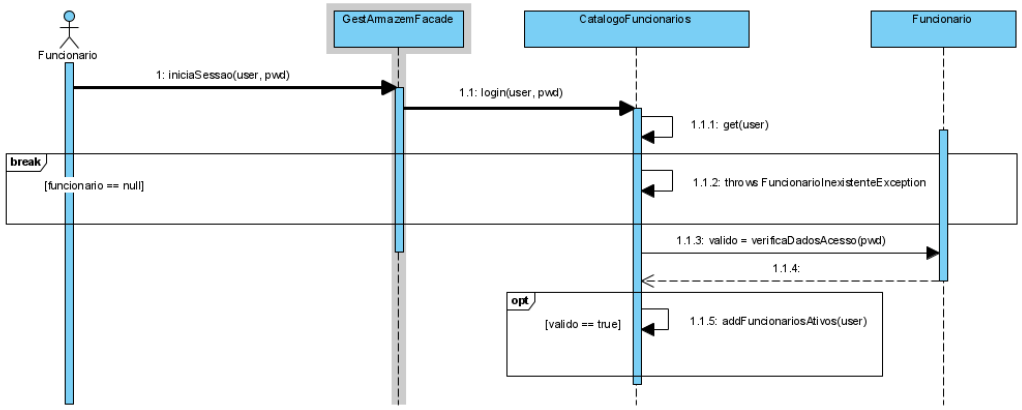
9. Verificar disponibilidade para descarga



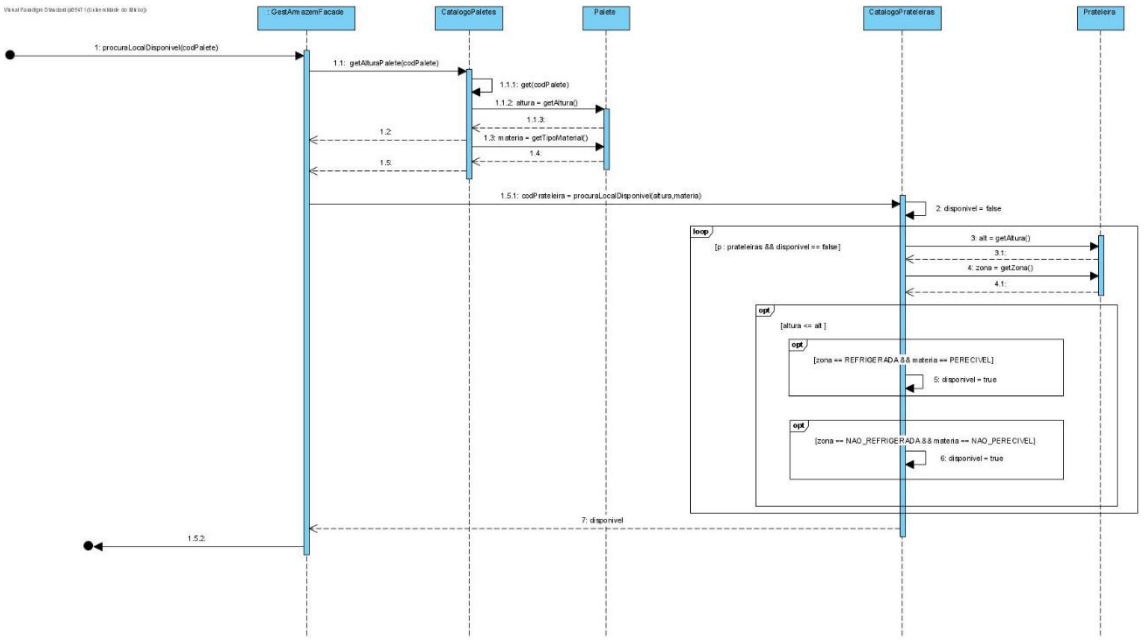
10. Verifica disponibilidade para recolha



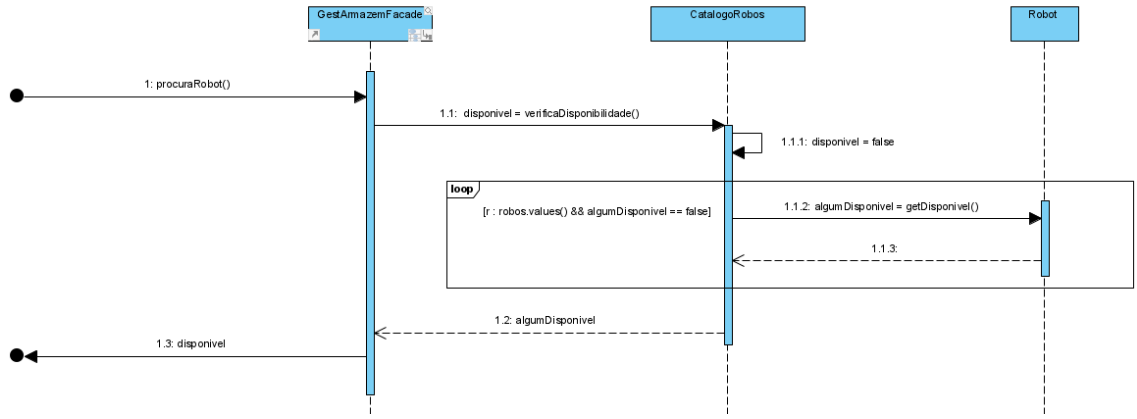
11. Iniciar sessão



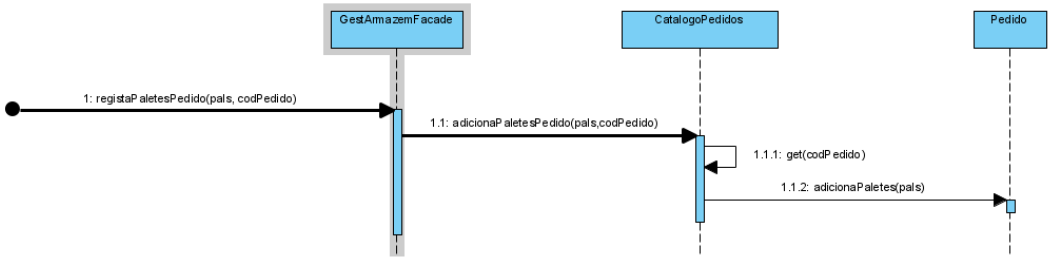
12. Procurar prateleira disponível



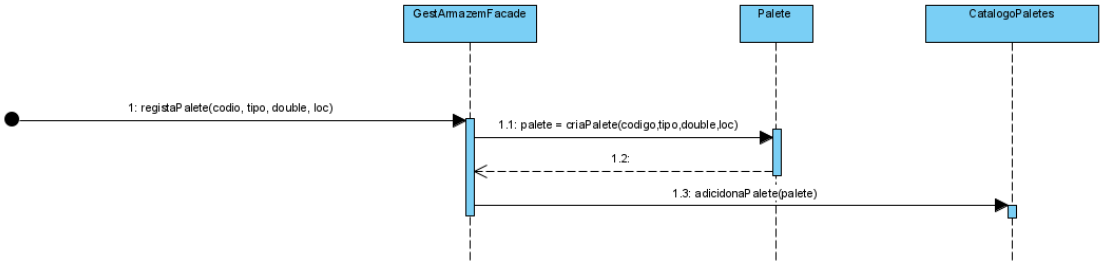
13. Procura robot



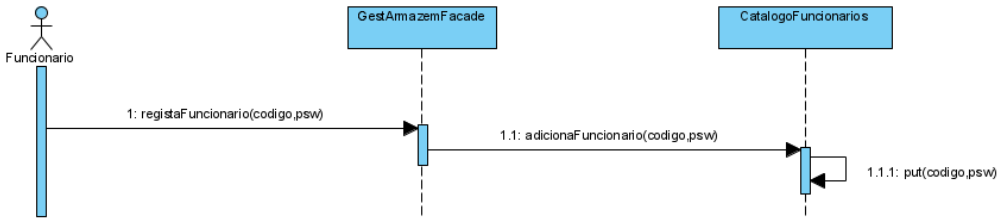
14. Regista palete



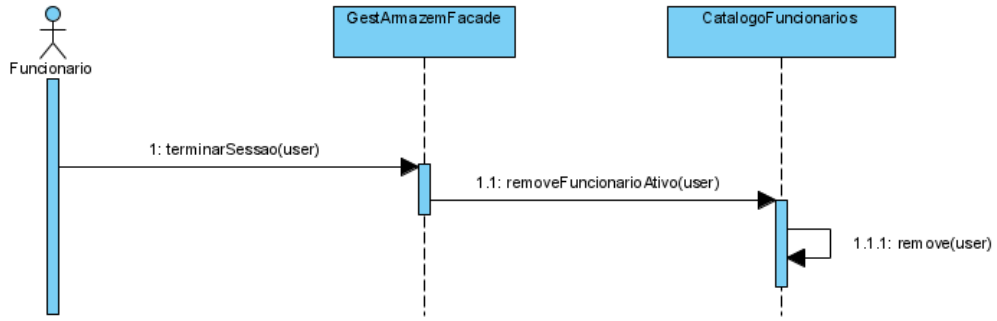
15. Regista paletes de pedido



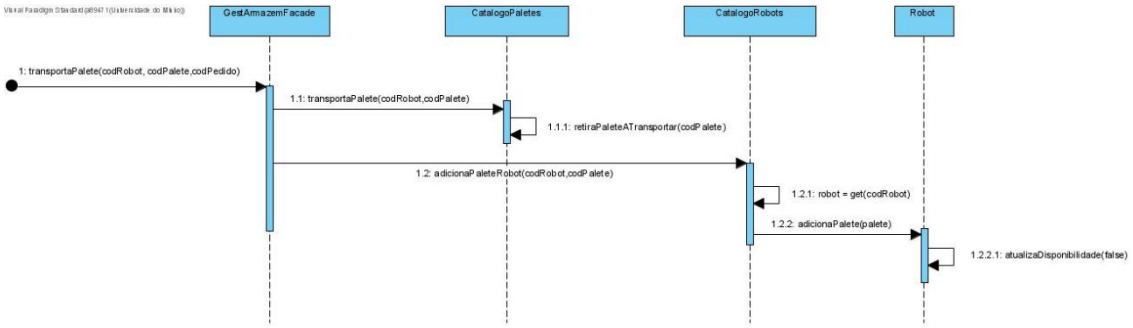
16. Regista funcionário



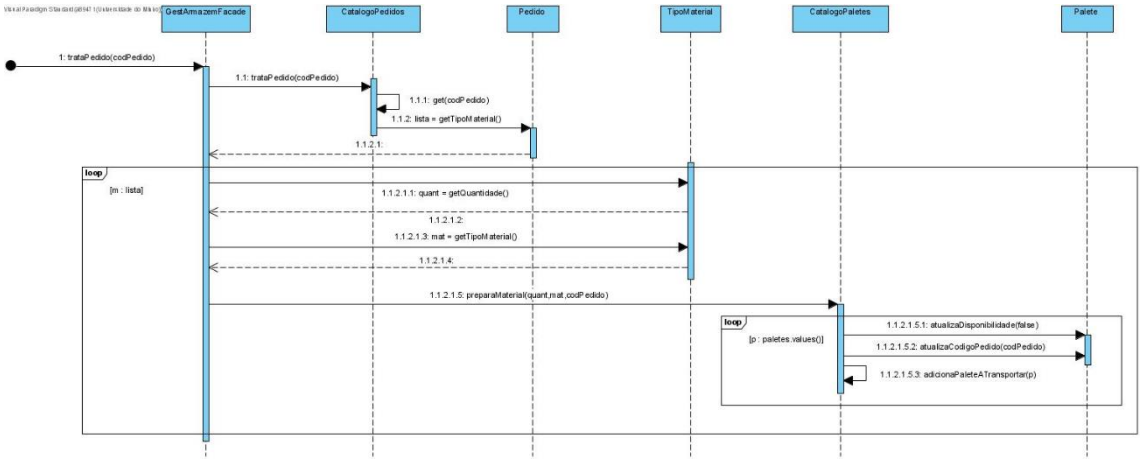
17. Terminar sessão



18. Transporta palete



19. Trata pedido



20. Valida disponibilidade de matérias requisitadas

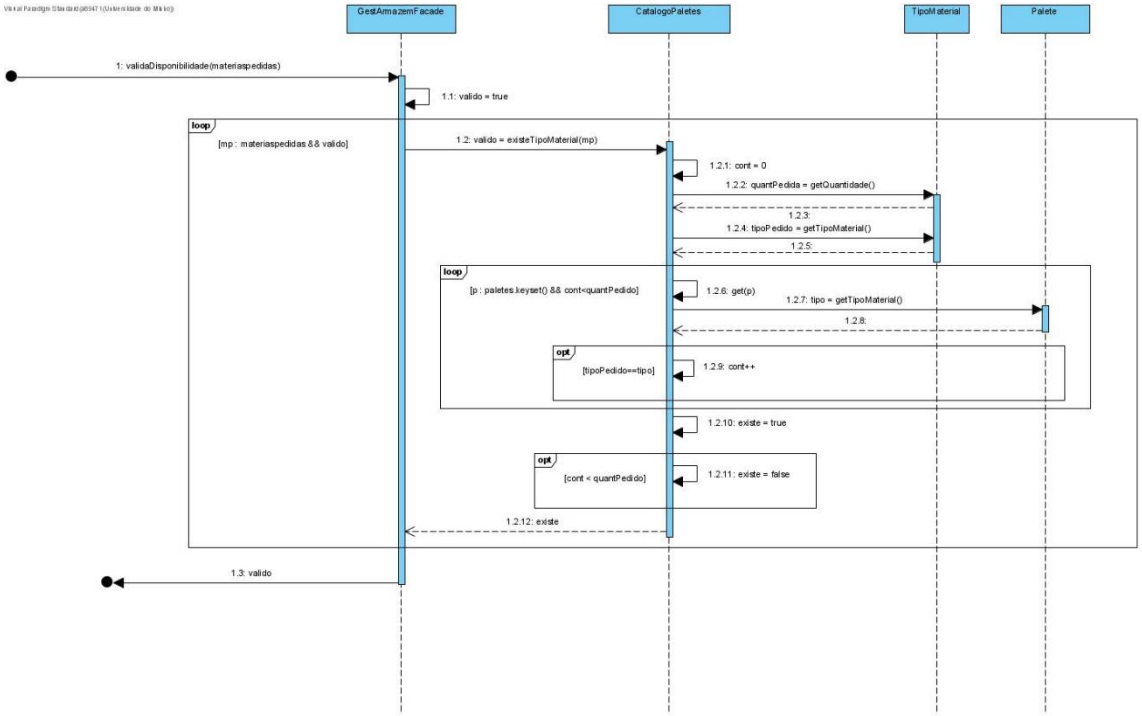


Diagrama de Classes

Com base no modelo de domínio foram selecionadas as componentes que dariam origem a classes, como por exemplo, a Palete e o Robot. Acrescido a estas foram também adicionadas algumas novas classes que implementam a lógica de negócio, como por exemplo, a classe Mapa e Localização. Após a análise de todos os diagramas de sequência desenvolvidos, bem como do diagrama de componentes onde estão definidos os subsistemas principais da modelação do nosso projeto, prosseguimos à realização do diagrama de classes, onde descrevemos as classes existentes, modeladas com os respetivos atributos e operações. Deste processo resultou a versão inicial do diagrama de classes:

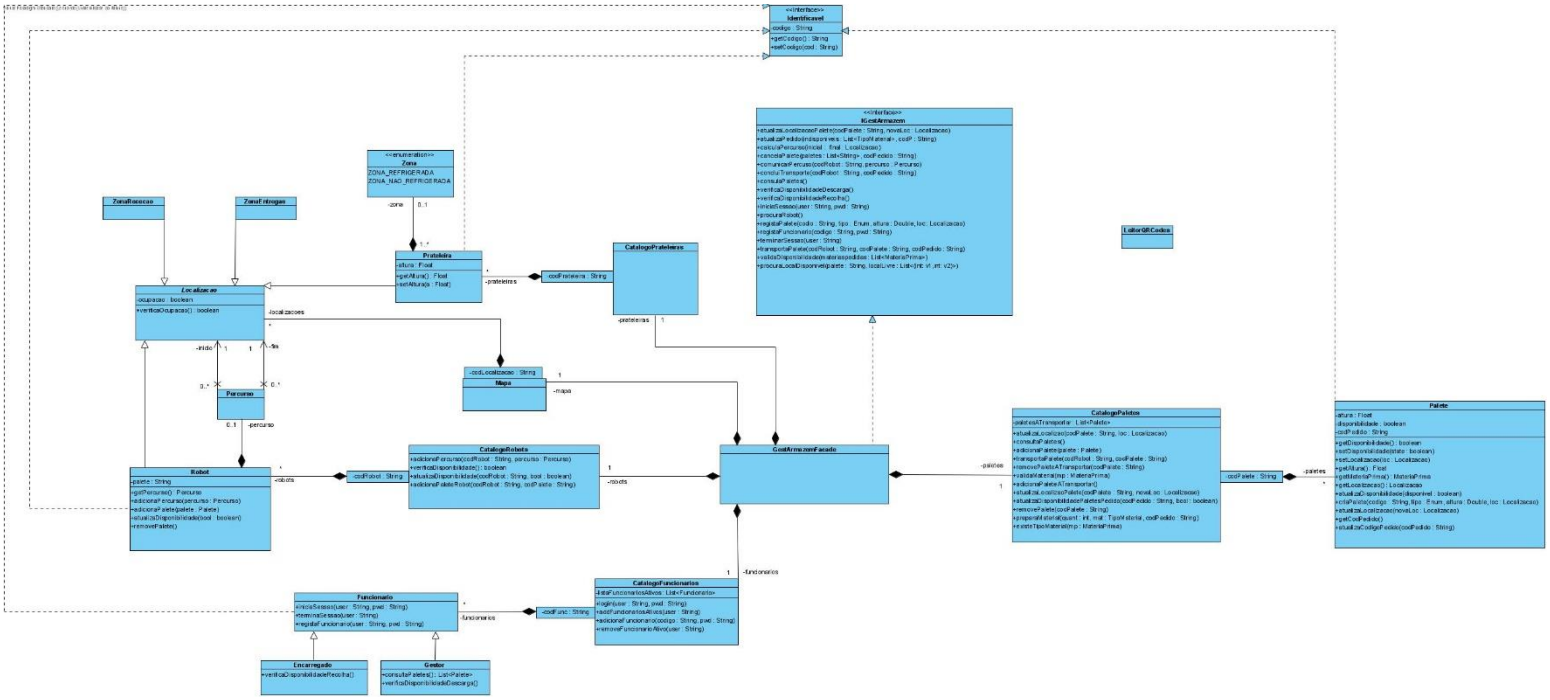


Figura 5 - Diagrama de Classes, versão inicial

No entanto, com o desenvolver do trabalho, e dada a natureza incremental adotada, viemos a atualizar este diagrama ao longo do projeto, o que deu resultado ao diagrama seguinte:

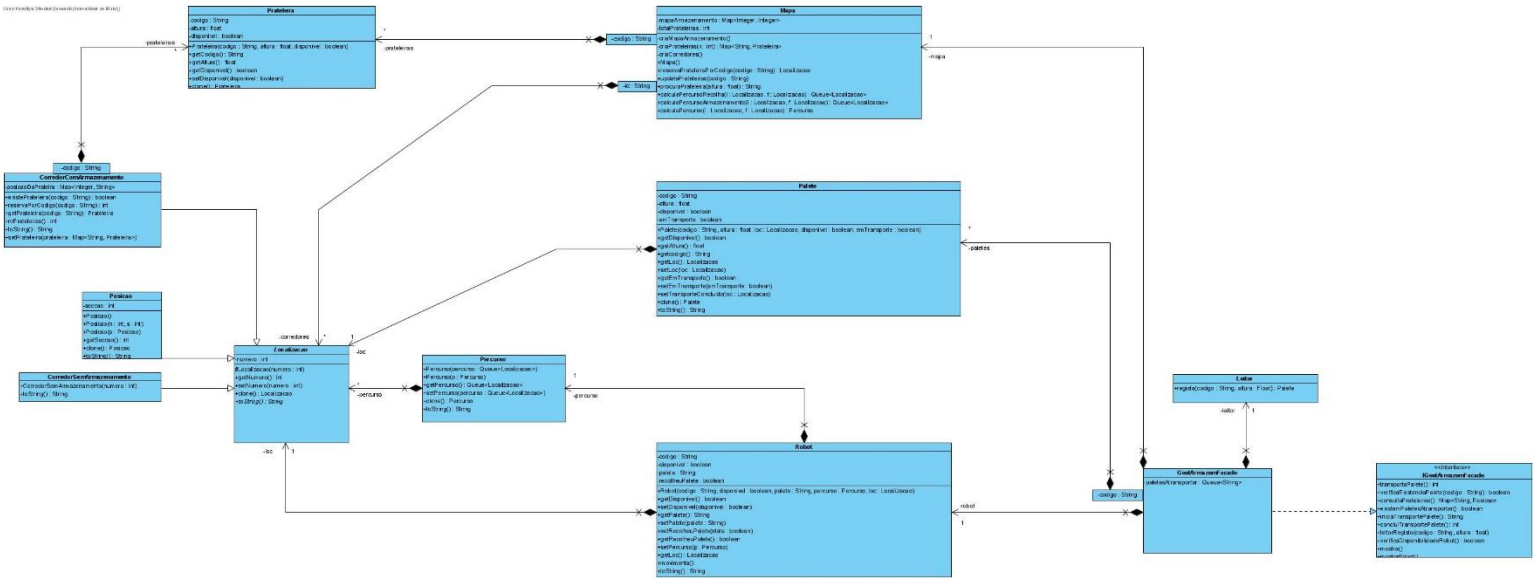


Figura 6 - Diagrama de Classes, versão final

Este diagrama tem como principais diferenças, a remoção das classes do tipo "Catálogo" que consideramos serem desnecessárias, pelo menos nesta fase do trabalho que apenas exigia um número bastante pequeno de métodos. Talvez fosse algo a considerar numa fase futura de forma a adicionar mais uma camada de entendimento e separação dos dados. Além disso foi feita uma renovação do pensamento relacionado com a Localização. Desta vez, a localização trata-se de uma classe abstrata que generaliza: uma Posição, um Corredor sem Armazenamento, ou um Corredor com Armazenamento. Apenas os corredores com armazenamento possuem Prateleiras. Além disso, é importante referir que apenas consideramos as classes que iriam ser implementadas nesta fase do desenvolvimento no diagrama de classes apresentado.

Diagrama de ORM

Após completo o diagrama de classes prosseguimos com o desenvolvimento do diagrama de Object Relational Mapping – ORM. Para isto, identificamos todas as entidades que tem estado, que tem necessidade de manter dados e, por isso, devem ser persistidas. Para além disso foram mapeados os relacionamentos, substituíram-se todas as coleções e maps por acessos à base de dados, recorrendo ao uso de DAO's.

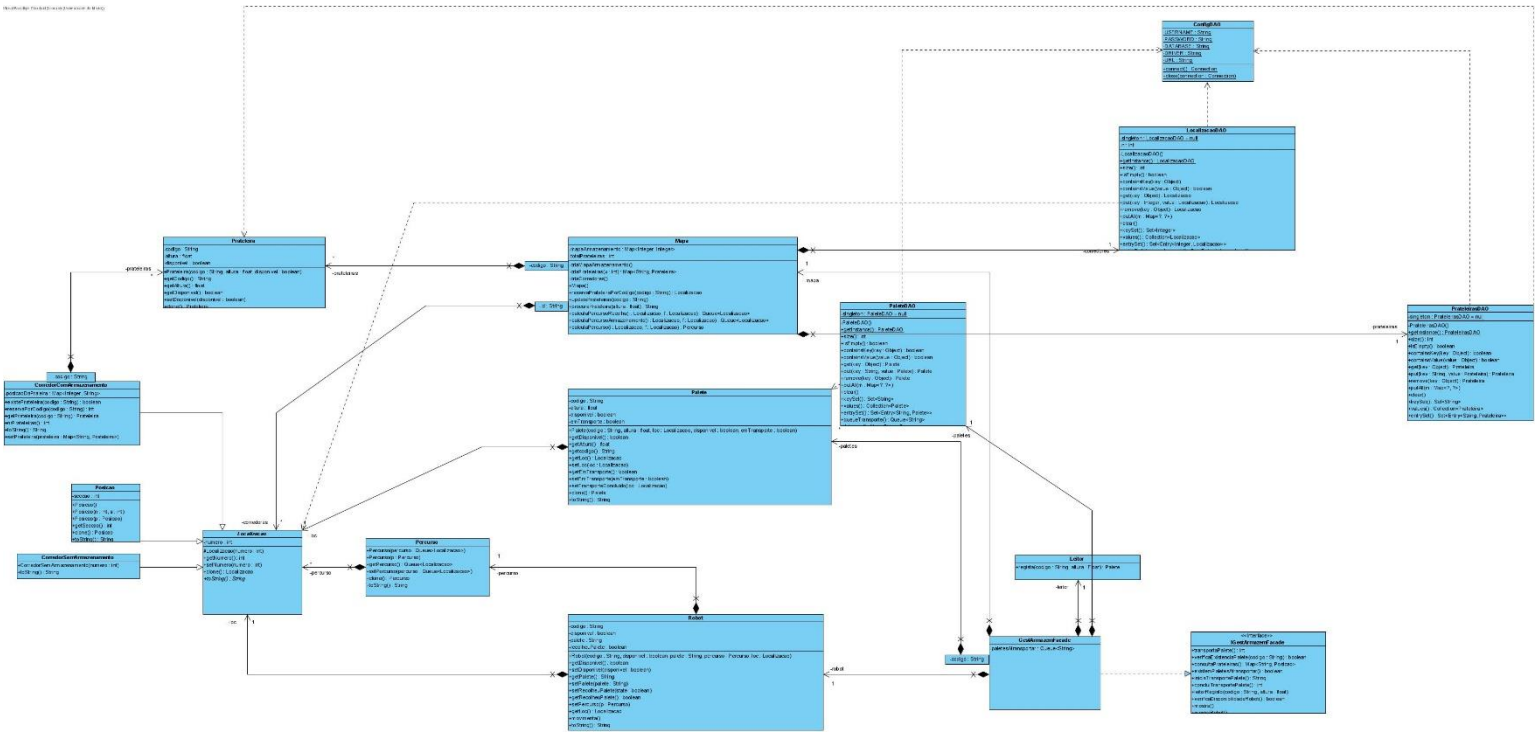


Figura 7 - Diagrama de ORM

Diagrama de Packages

À medida que os projetos se tornam mais complexos, é útil implementar um diagrama de packages para ilustrar a arquitetura de um sistema e as dependências das suas classes. Na criação dos packages dividimos as responsabilidades em quatro grupos: o que agrupa as classes

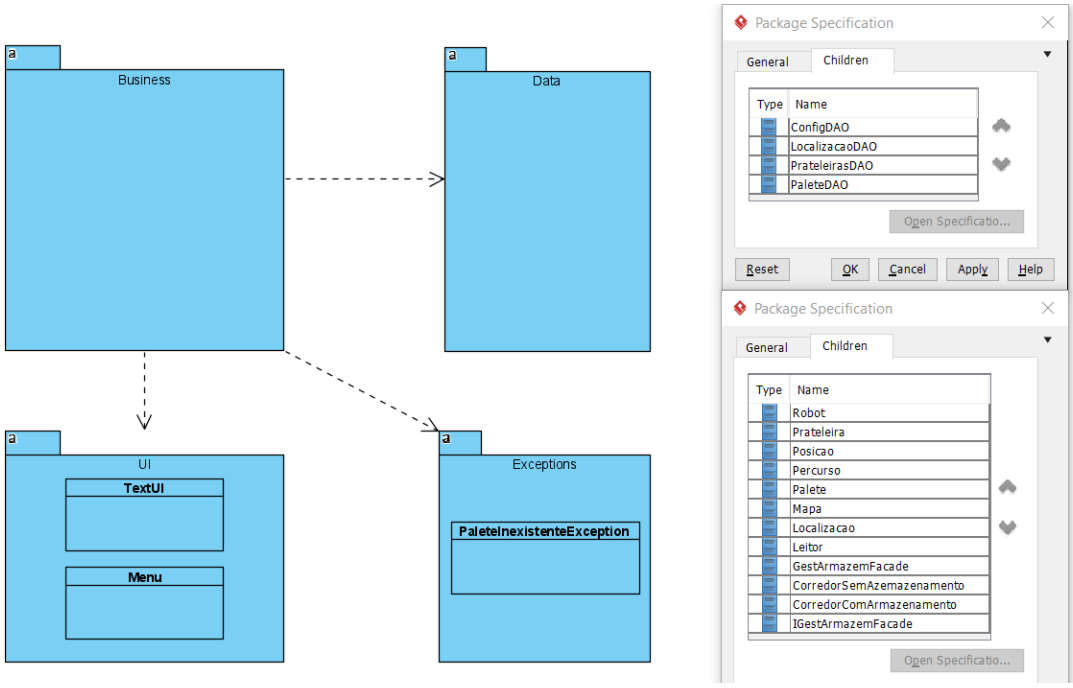


Figura 8 - Diagrama de Packages

que interagem com os atores do sistema, o que agrupa as classes relacionadas com a parte gráfica, o que agrupa as exceções e o que agrupa os Dados armazenados em base de dados.

Assim o manuseamento será muito mais organizado quando for necessário tratar de um use case específico.

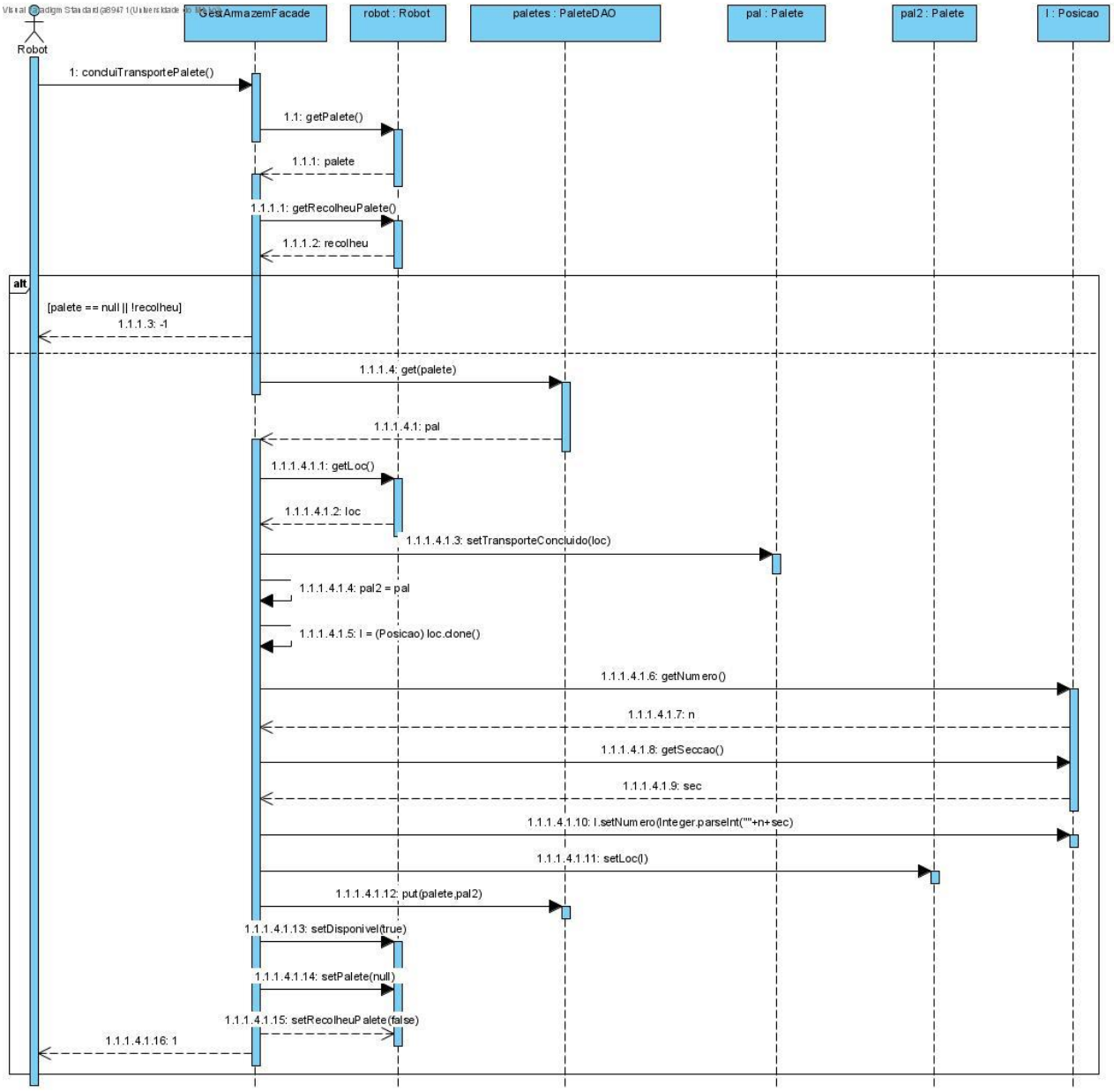
Diagrama de Sequência de Implementação

No seguimento dos diagramas de sequência de subsistemas, e já na terceira fase, foram desenvolvidos os diagramas de sequência de implementação, que se distinguem por ser uma refinação dos anteriores. Nesta fase já se faz referência a classes e às suas instâncias específicas, é também já utilizada uma sintaxe muito próxima da que foi usada no código Java.

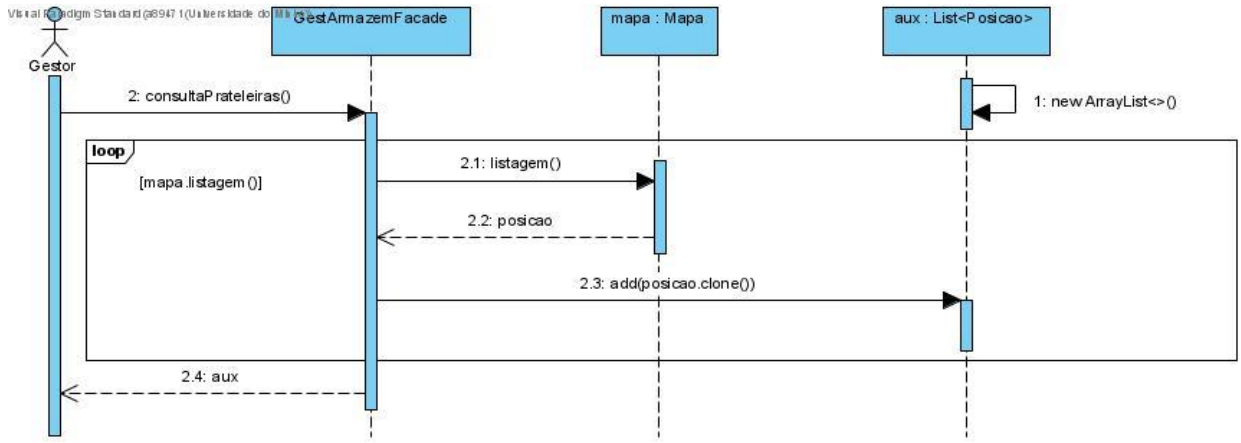
Uma vez que estes diagramas foram começados já na fase de implementação, apenas consistem nos casos pretendidos considerar nesta fase. Além disso, à medida que era implementado o projeto, estes diagramas foram sendo atualizados pelo que está tudo bem definido, distribuído e coerente com aquilo que foi implementado.

É importante também referir que temos consciência que existem algumas disparidades entre estes diagramas de sequência e os apresentados anteriormente isto justificou-se na medida em que as soluções pensadas na segunda fase foram completamente repensadas o que levou a uma restruturação bastante grande do problema, ainda assim, a ideia geral foi conservada.

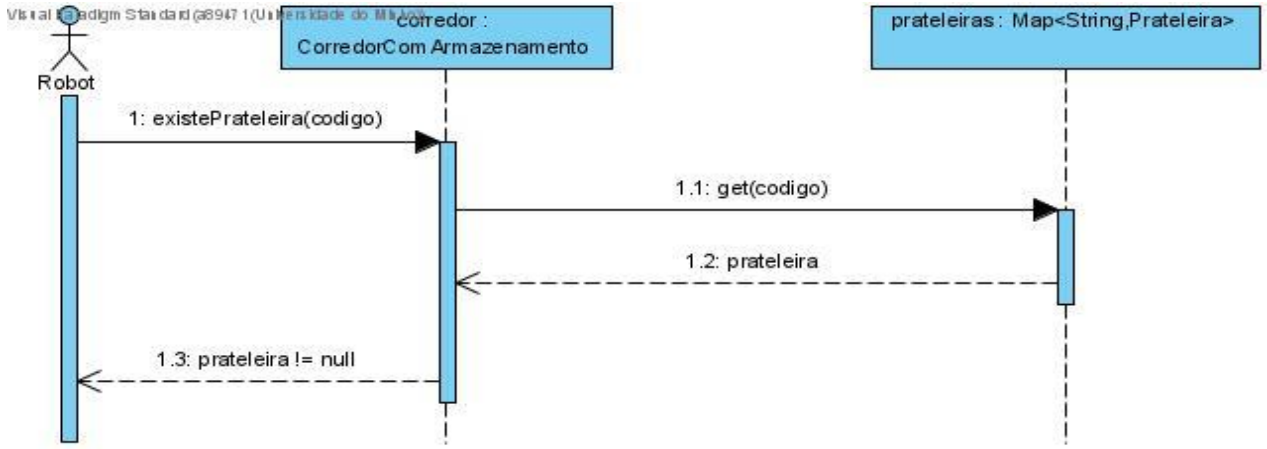
1. concluiTransportePalete



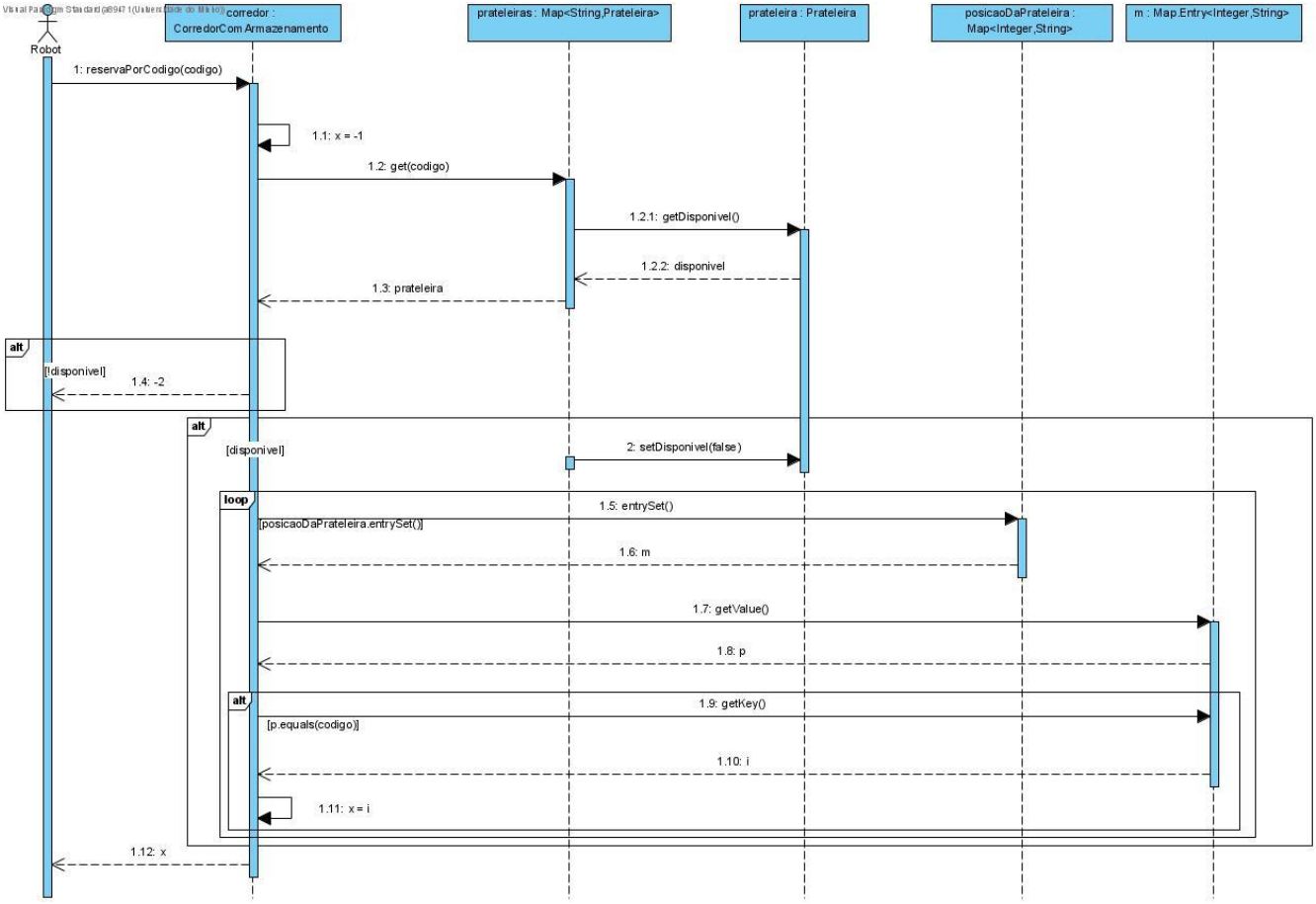
2. consultaPrateleiras



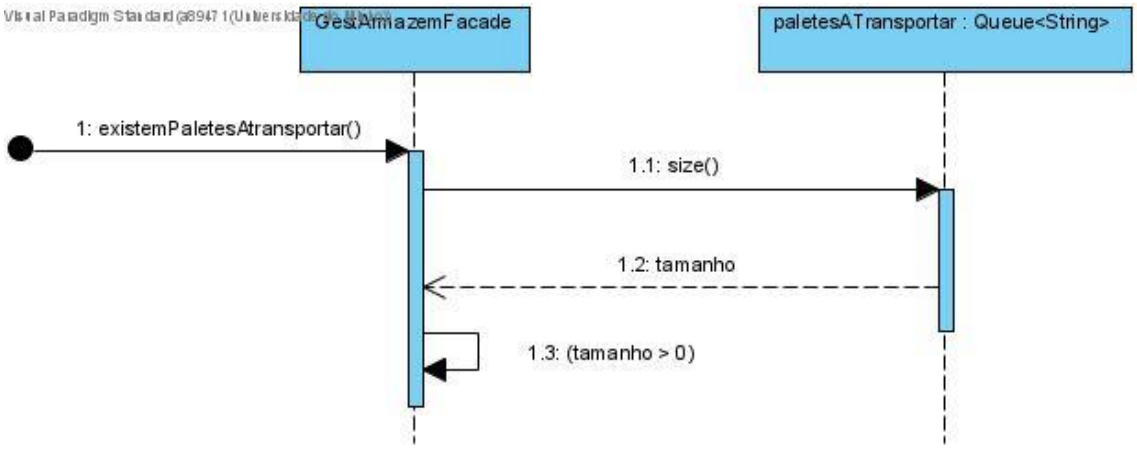
3. existePrateleira



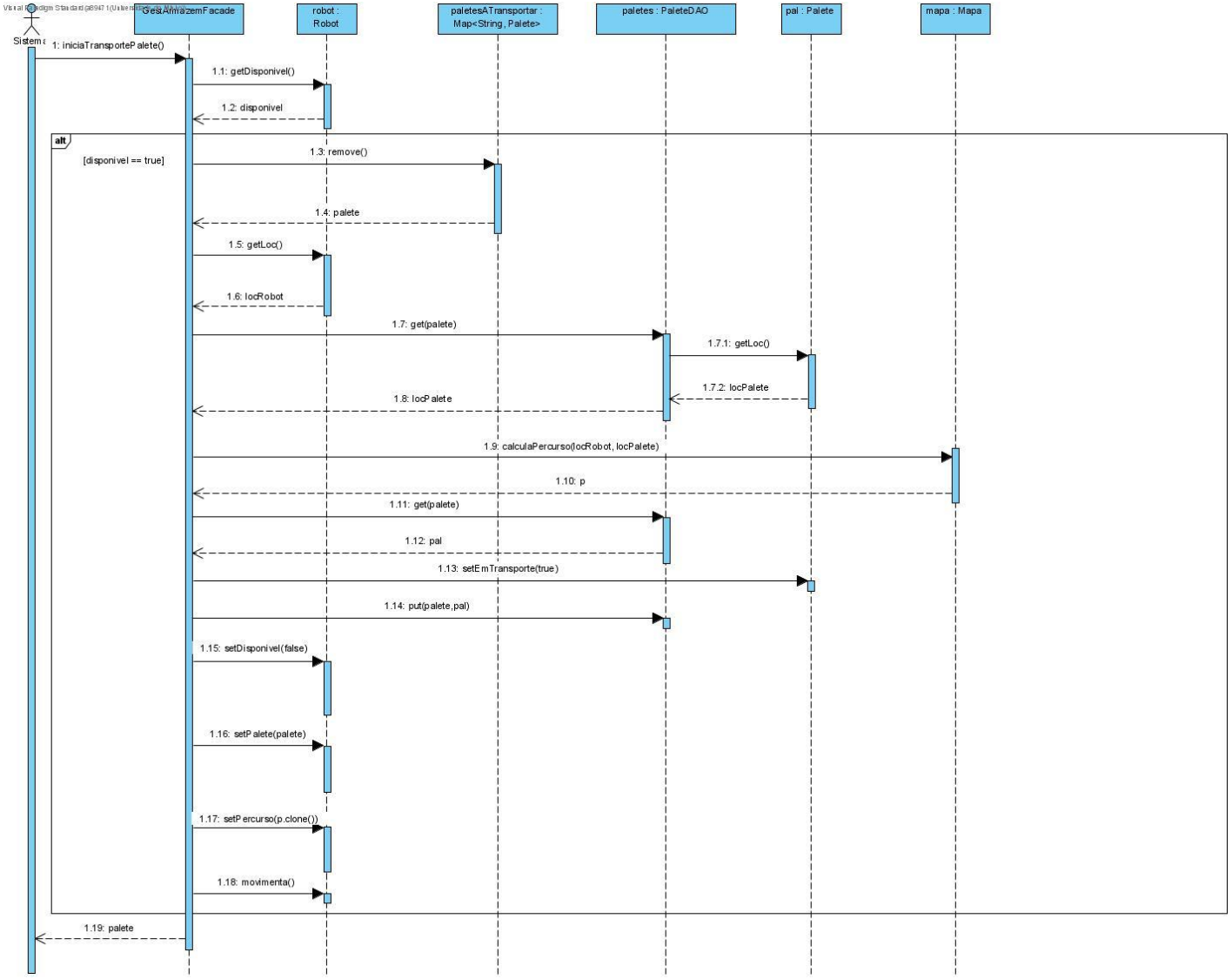
4. reservaPorCodigo



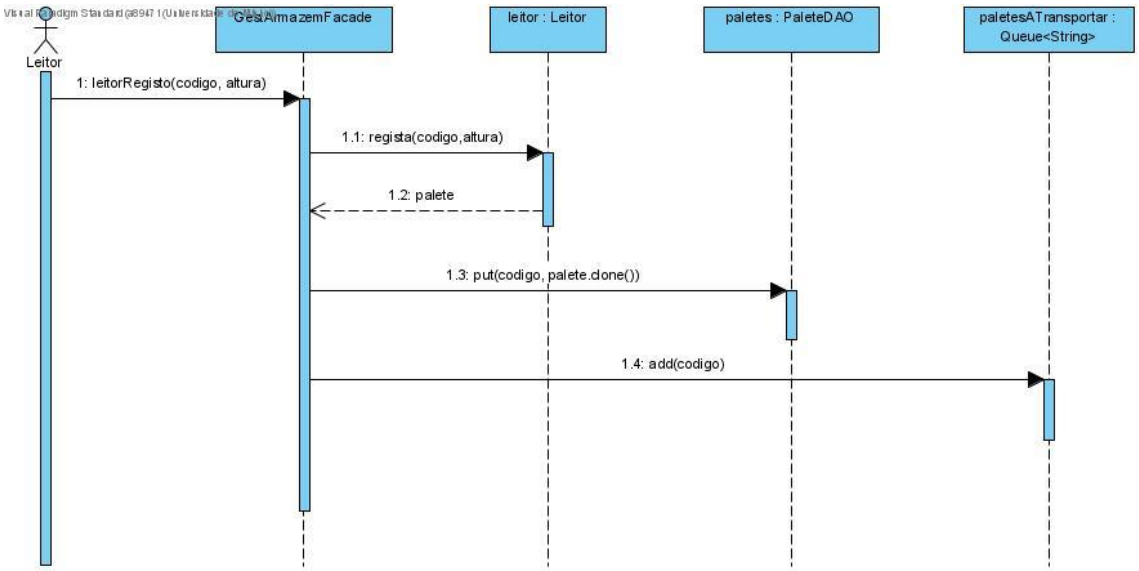
5. existemPaletesATransportar



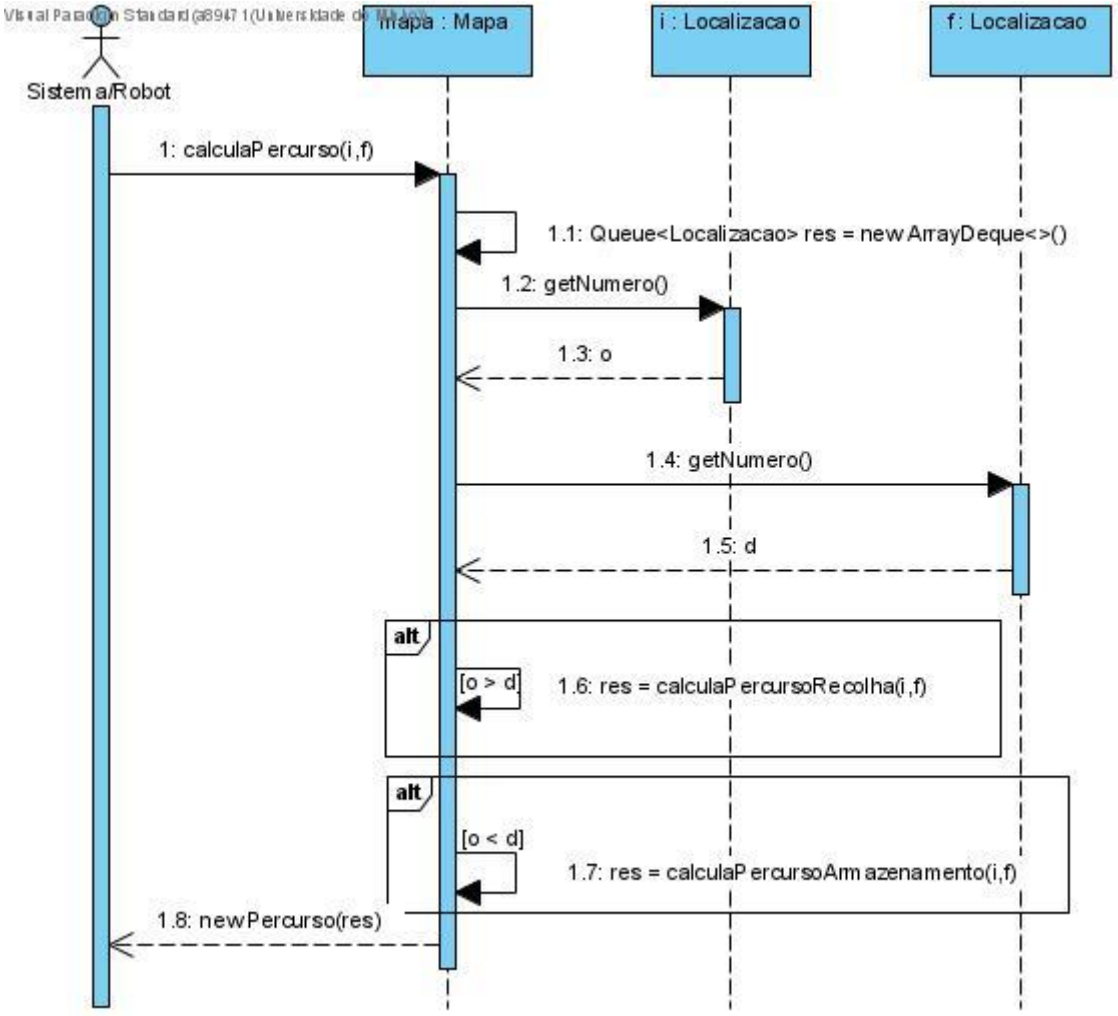
6. iniciaTransportePalete



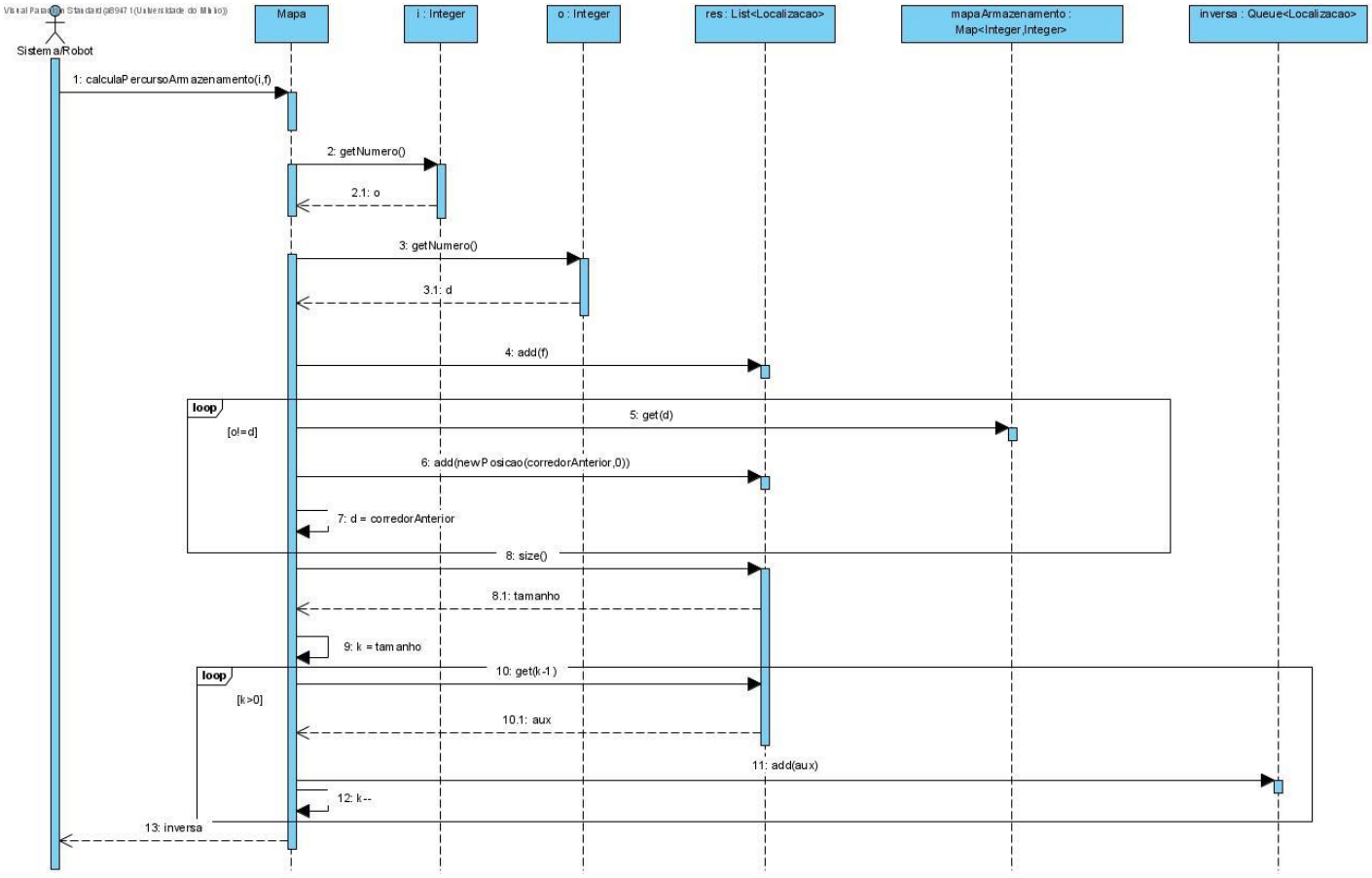
7. leitorRegisto



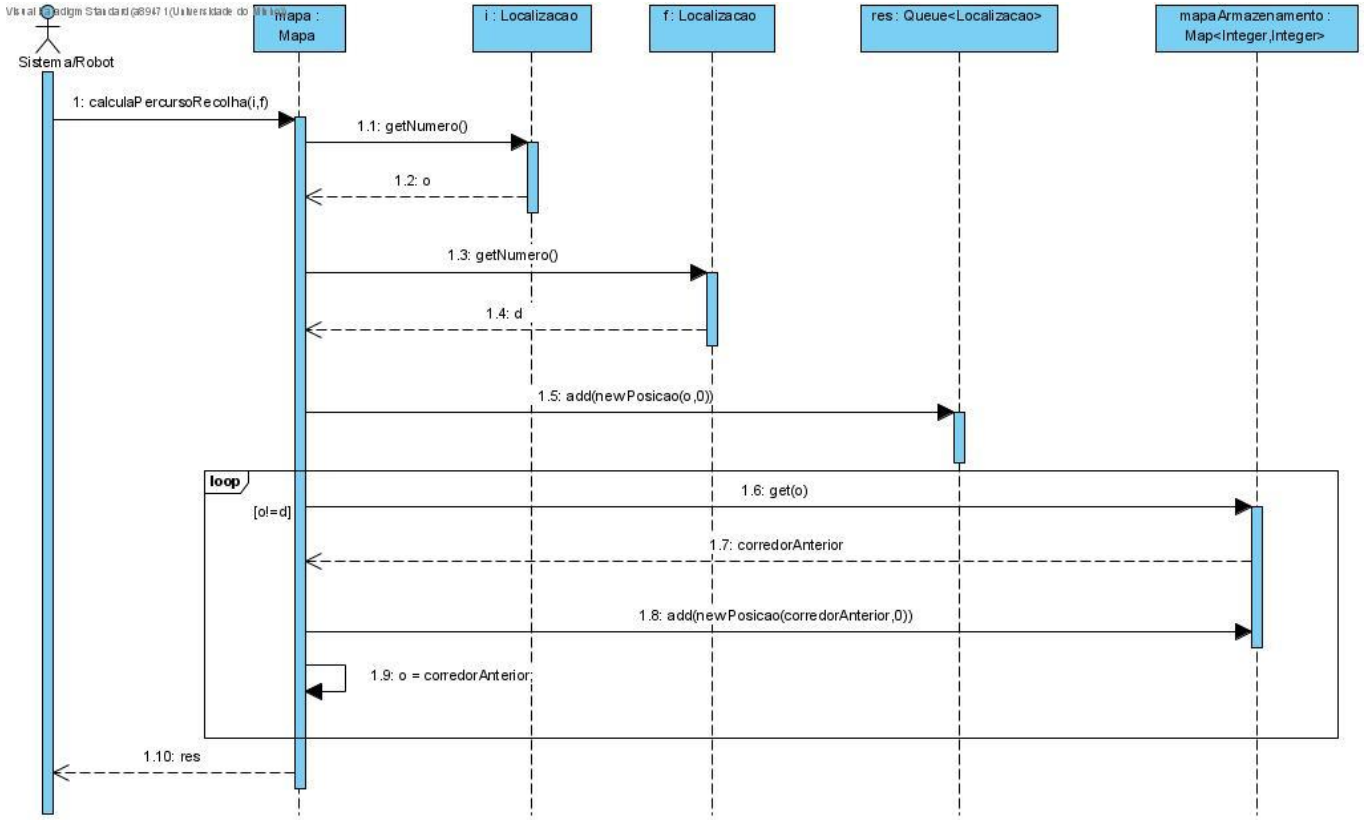
8. calculaPercurso



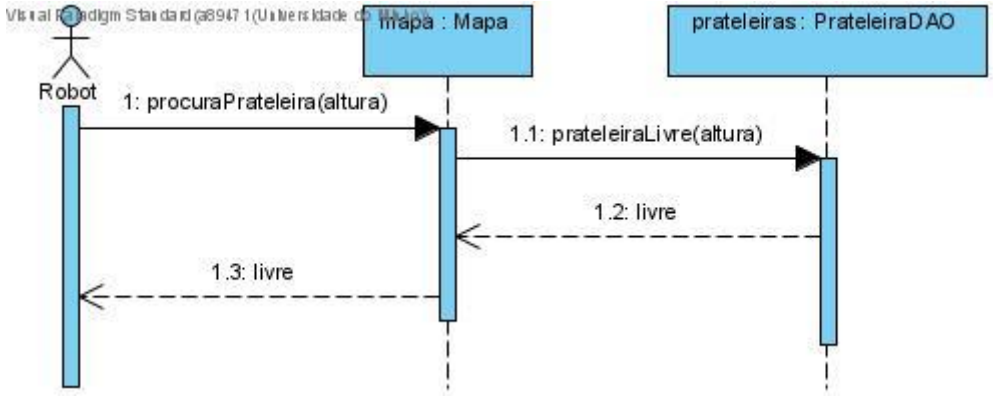
9. calculaPercursoArmazenamento



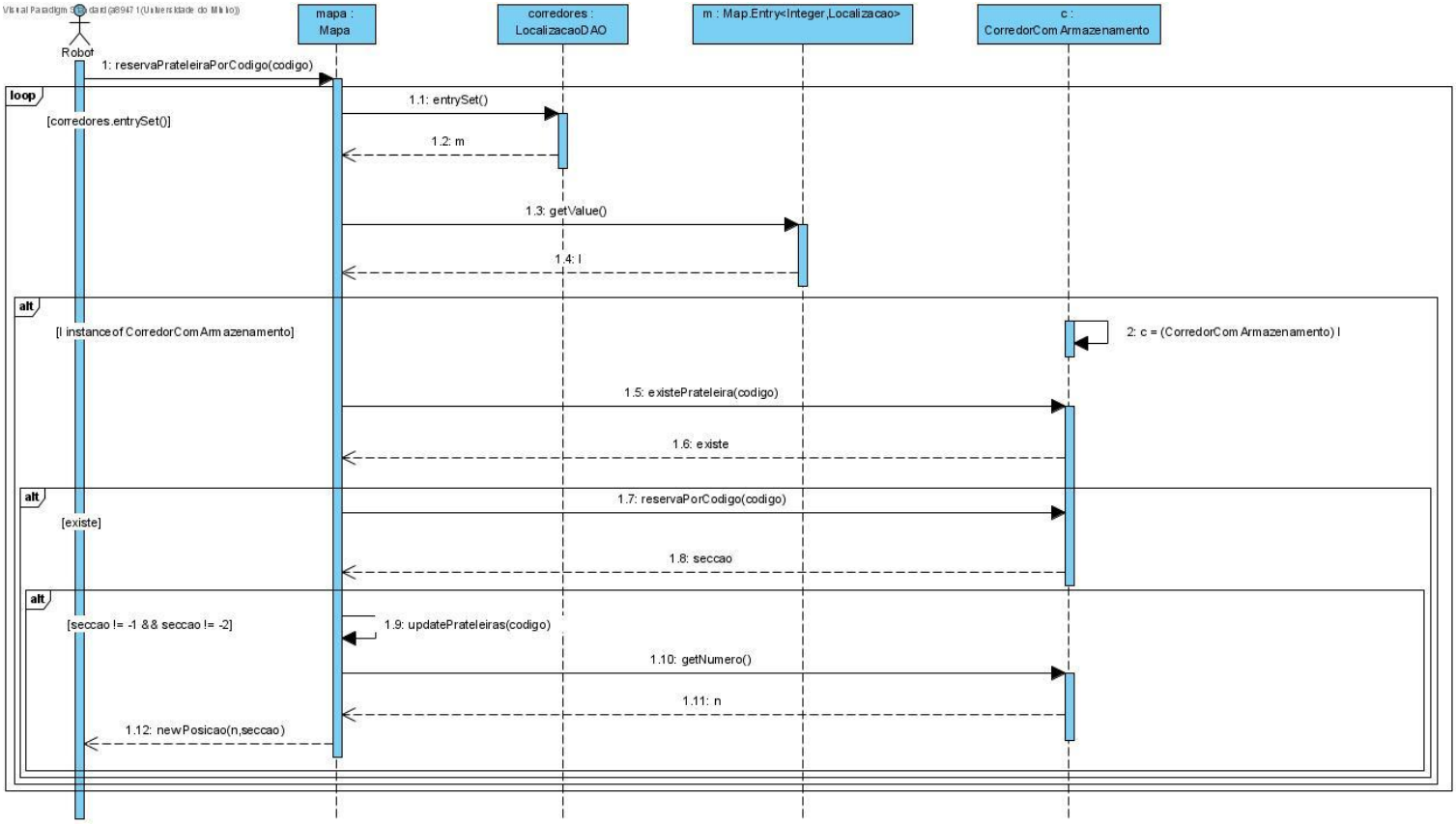
10. calculaPercursoRecolha



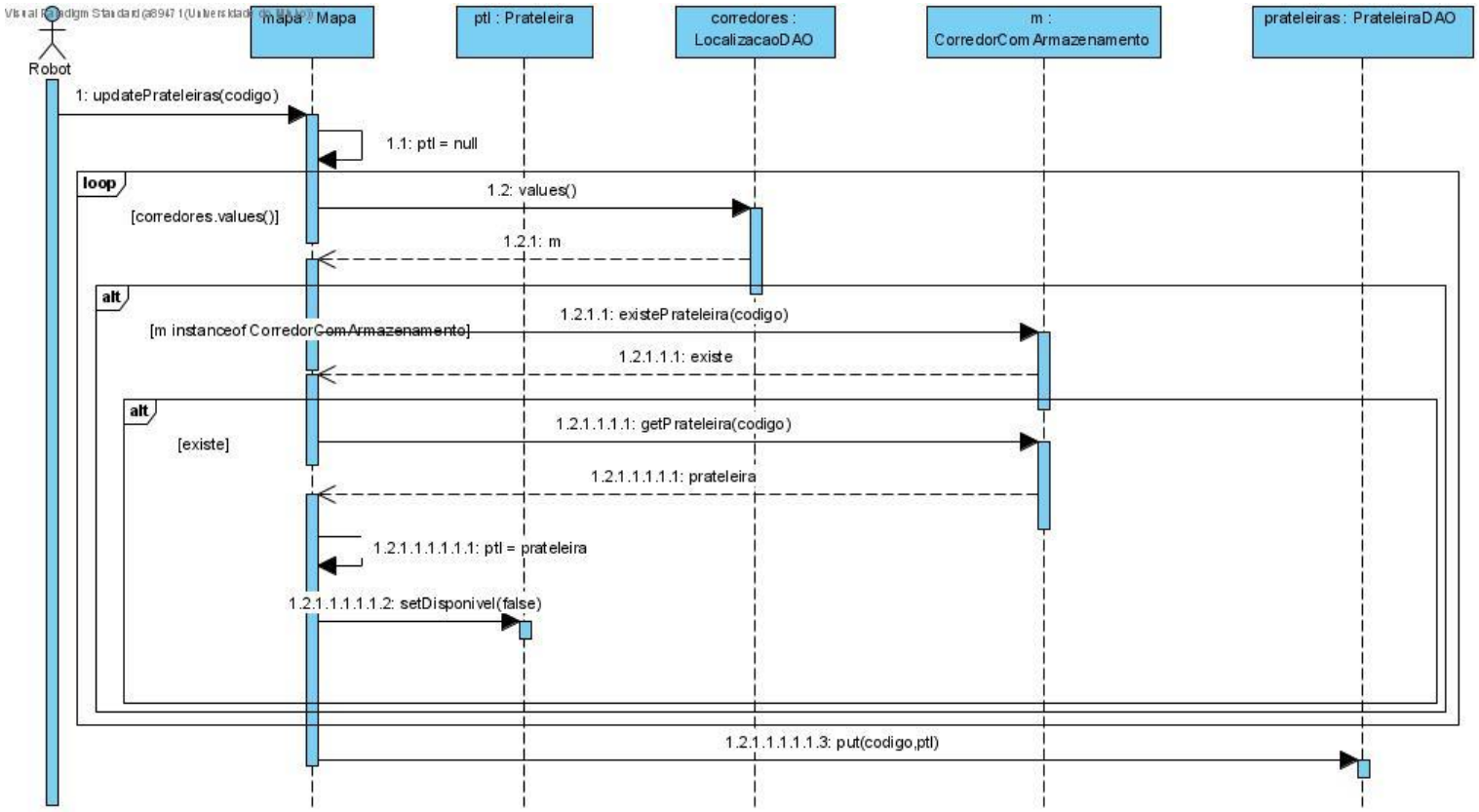
11. procuraPrateleira



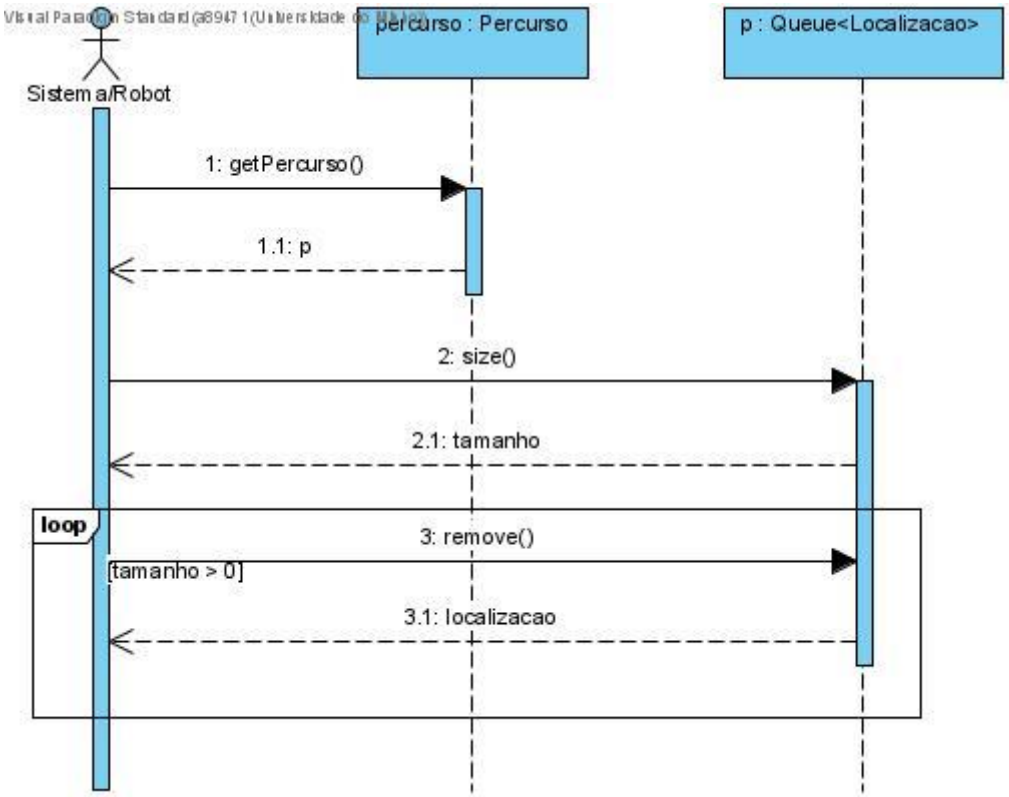
12. reservaPrateleiraPorCodigo



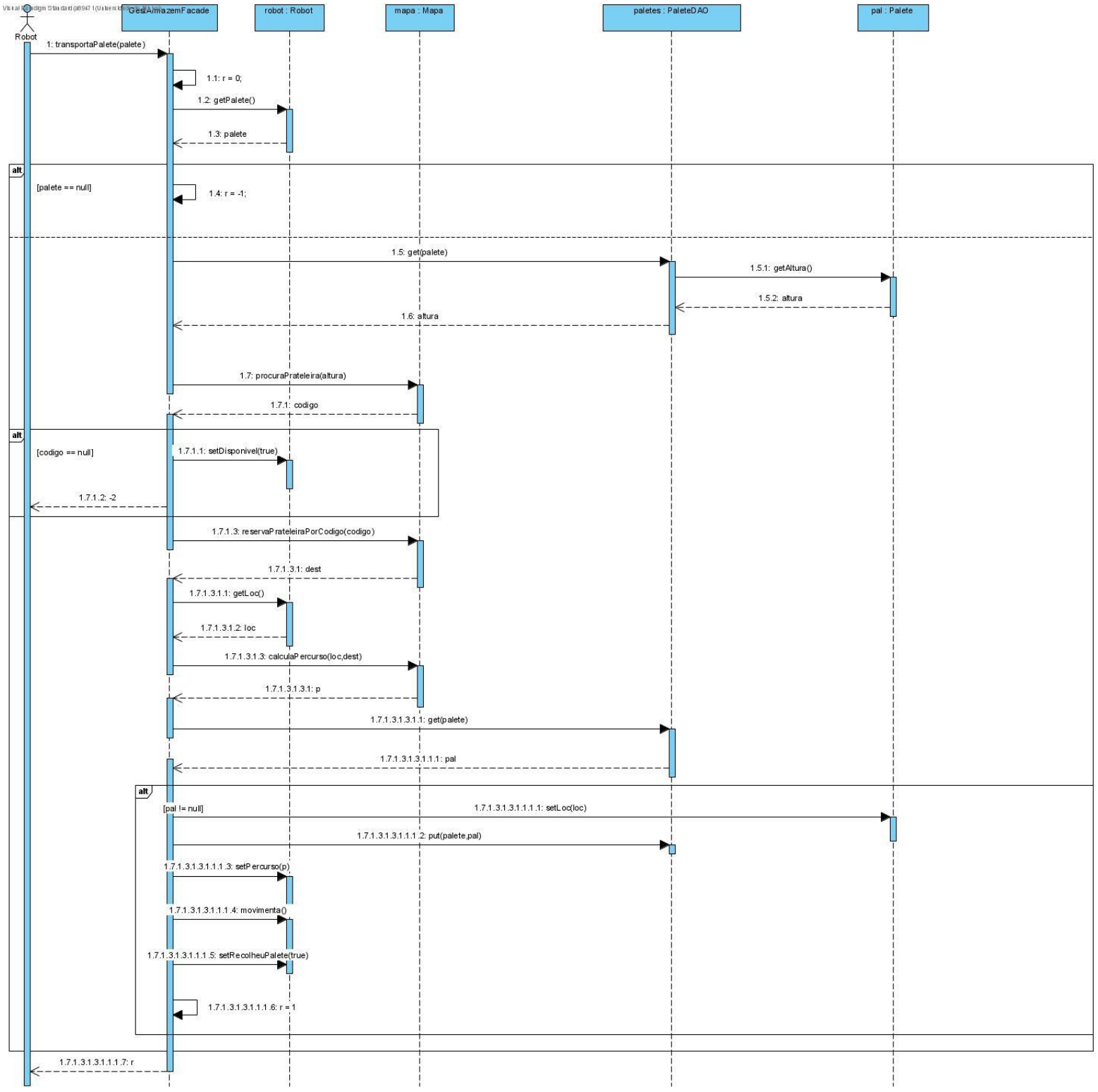
13. updatePrateleiras



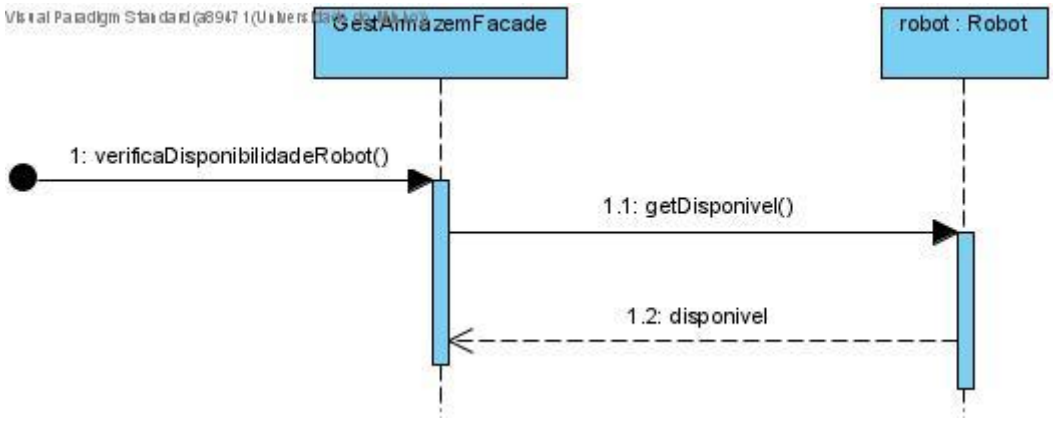
14. movimenta



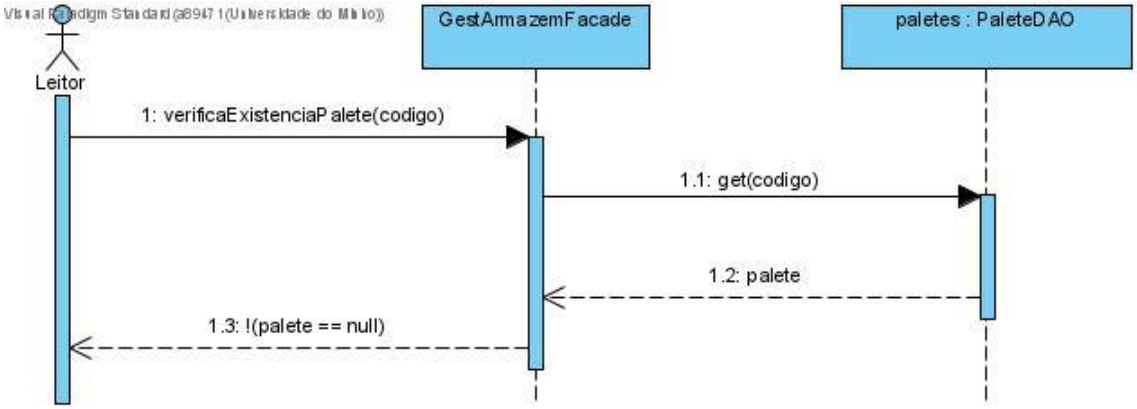
15. transportaPalete



16. verificaDisponibilidadeRobot



17. verificaExistenciaPalete



Base de Dados

De forma a deixarmos de guardar os dados em memória, procedemos à criação de uma base de dados que servirá para armazenar todas as informações necessárias para o bom funcionamento do nosso software. Para tal, utilizamos o MySQL. Abaixo, podemos ver o modelo lógico da base de dados.

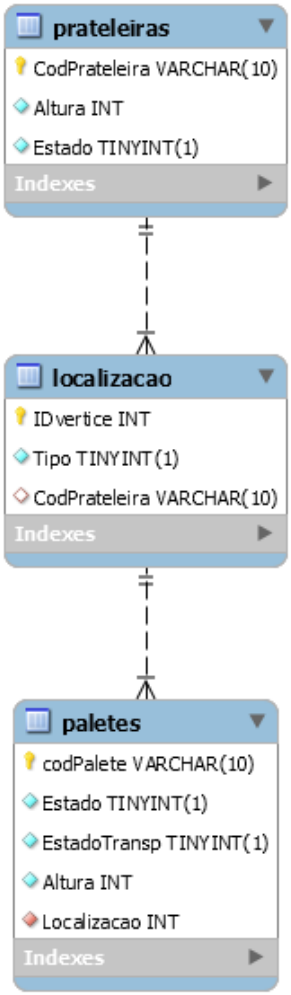


Figura 9 - Modelo Lógico da Base De Dados.

Implementação

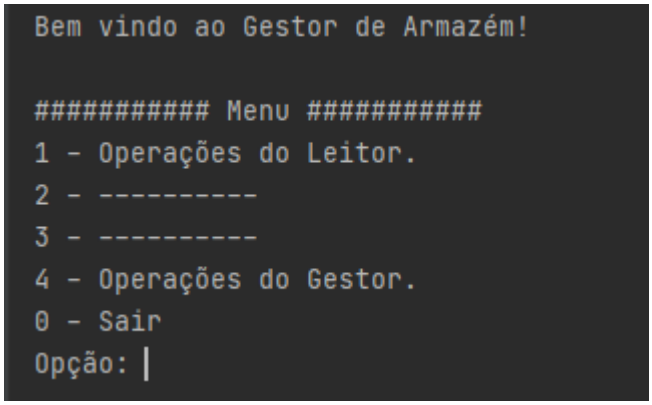
Finalmente, e depois de acabado todo o processo de formulação do projeto decidimos passar à implementação deste na linguagem de programação JAVA, utilizando o IDE IntelliJ.

Numa primeira ocasião, verificamos pelo diagrama de Package que deveríamos ter quatro packages na nossa aplicação. Decidimos então começar a implementação pela camada de negócio, deste modo, utilizamos os Use Cases e os diagramas de sequência para nos orientarmos. Efetivamente, com os diagramas de classe desenvolvidos, todo o processo se tornou mais intuitivo, pois identificamos como e quais os métodos que eram necessários serem desenvolvidos. No entanto, utilizamos em certos casos métodos de classe que facilitaram o seu acesso pelo que alguns aspetos dos diagramas se tornaram um pouco redundantes.

É importante ainda salientar que usamos o MySQL Workbench para a criação do modelo de base de dados que nos pareceu ser o mais adequado.

De seguida, exemplificamos a execução do programa com uma tarefa simples de registar uma palete no sistema e posteriormente ela ser guardada no armazém.

No Menu inicial dividimos as operações que cada ator pode executar (leitor de códigos, sistema, robot e gestor) em diferentes menus. Em cada operação executada no sistema, o menu principal é atualizado para as operações que cada entidade consegue realizar num dado momento. Numa primeira execução do programa, conseguimos ver que apenas o leitor ou o gestor conseguem realizar ações, pois o robot só funciona como transportador de paletes, e ainda não as há no armazém, e o sistema só funciona como ditador da necessidade de transportar alguma palete, e como dito, ainda não há paletes registadas.



Comecemos por registar uma palete no sistema, escolhendo a opção 1 no menu anterior. De seguida, é apresentado o seguinte menu e executada a opção 1.

```
########### Menu ###########
1 - Registar palete.
0 - Sair
Opção: 1
Codigo da palete: 1
Altura da palete: 10
 -> Palete registada no sistema com sucesso.
 -> Palete adcicionada à lista de paletes a transportar.
Palete{codigo='1', altura=10.0, loc=Número: 0 Posição { secção = 0}, disponivel=false, emTransporte=false}
Robot { código = '1', disponivel = true, palete = 'null', percuso = null, localização = Número: 0}
```

Como é possível verificar, inserindo o código da palete e a respetiva altura, esta fica registada no sistema, caso ainda não haja nenhuma registada com o mesmo código.

Voltando ao menu inicial, podemos verificar que o sistema já consegue realizar operações, pois já existe uma palete registada.

```
########### Menu ###########
1 - Operações do Leitor.
2 - Operações do Sistema.
3 - ----------
4 - Operações do Gestor.
0 - Sair
Opção:
```

Escolhendo a opção 2, e posteriormente, no menu do sistema, a opção 1, é possível notificar o robot que existe uma palete a necessitar de transporte. Neste caso, o robot, caso esteja disponível, dirige-se ao local onde a palete se encontra.

```
########### Menu ###########
1 - Operações do Leitor.
2 - Operações do Sistema.
3 - ----------
4 - Operações do Gestor.
0 - Sair
Opção: 2

########### Menu ###########
1 - Preparar transporte de palete.
0 - Sair
Opção: 1
 -> Robot notificado!
 -> Palete selecionada: 1
Palete{codigo='1', altura=10.0, loc=Número: 0 Posição { secção = 0}, disponivel=false, emTransporte=true}
Robot { código = '1', disponivel = false, palete = '1', percuso = Percurso { [] }, localização = Número: 0}
```

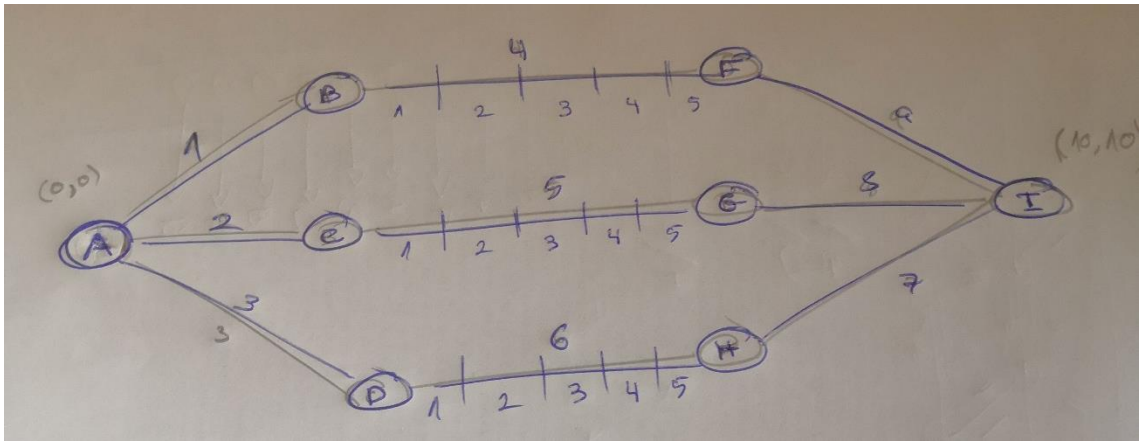
Voltando ao menu inicial, podemos de novo verificar que desta vez o robot já pode executar ações, pois já existem paletes para ele transportar.

```
############ Menu ############
1 - Operações do Leitor.
2 - ----------
3 - Operações do Robot.
4 - Operações do Gestor.
0 - Sair
Opção:
```

Selecionando o menu do robot, e escolhendo a opção de notificar que a palete foi recolhida, obtemos o seguinte:

```
############ Menu ############
1 - Notificar recolha da palete.
2 - Notificar entrega da palete.
0 - Sair
Opção: 1
A percorrer -> Número: 0 Posição { secção = 0}
A percorrer -> Número: 1 Posição { secção = 0}
A percorrer -> Número: 4 Posição { secção = 0}
 -> Palete recolhida e transportada com sucesso!
Palete{codigo='1', altura=10.0, loc=Número: 0 Posição { secção = 0}, disponivel=false, emTransporte=true}
Robot { código = '1', disponivel = false, palete = '1', percuso = Percurso { [Número: 0 Posição { secção = 0}, Número: 1 Posição { secção = 0}, Número: 4
```

Aqui é mostrado o percurso que o robot faz dentro do armazém de forma a armazenar a palete no sítio indicado. O grafo que representa a planta do armazém é o seguinte:



Se posteriormente escolhermos a opção 2 do menu do robot, notificamos que a palete foi entregue no local de armazenamento com sucesso.

```
########### Menu ###########
1 - Notificar recolha da palete.
2 - Notificar entrega da palete.
0 - Sair
Opção: 2
 -> Palete entregue!
Palete{codigo='1', altura=10.0, loc=Número: 40 Posição { secção = 0}, disponivel=true, emTransporte=false}
Robot { código = '1', disponivel = true, palete = 'null', percuso = Percurso { [Número: 0 Posição { secção = 0}, Número: 1 Posição { secção = 0}, Número: 4
```

De seguida, para teste, se voltarmos a selecionar a opção de notificar a recolha da palete, vemos que aparece uma mensagem a dizer que o robot não tem paletes a entregar.

```
########## Menu ##########
1 - Notificar recolha da palete.
2 - Notificar entrega da palete.
0 - Sair
Opção: 1
 -> O robot não tem paletes a recolher!
Palete{codigo='1', altura=10.0, loc=Número: 40 Posição { secção = 0}, disponivel=true, emTransporte=false}
Robot { código = '1', disponivel = true, palete = 'null', percuso = Percurso { [Número: 0 Posição { secção = 0}, Número: 1 Posição { secção = 0}, Número: 4
```

Voltando ao menu inicial, estando já todas as paletes do sistema armazenadas, tanto o robot como o sistema não podem executar mais ações. No entanto, nesta altura, podemos solicitar ao gestor a listagem das paletes registadas no sistema:

```
########### Menu ###########
1 - Operações do Leitor.
2 - ----------
3 - ----------
4 - Operações do Gestor.
0 - Sair
Opção: 4

########### Menu ###########
1 - Consultar listagem de localizações ocupadas.
0 - Sair
Opção: 1
Palete: 1-> Corredor: 4-> Prateleira: 1
```

Como foi possível verificar, a implementação é clara e de fácil compreensão, para além de ser bastante prática de utilizar.

Análise Crítica

A primeira fase do projeto teve como objetivo a análise detalhada dos requisitos, foram desenvolvidos um modelo de domínio e um modelo de Use Cases. O modelo de domínio revelou ser essencial na medida em que nos permitiu ter uma visão mais clara e abstrata da realidade e permitiu a identificação correta dos diferentes papéis nas relações entre as entidades. Em relação ao modelo de Use Cases, conseguimos entender melhor os requisitos do Sistema, isto é, o que o Sistema deve fazer e como, prevenindo deste modo a implementação de funcionalidades que não sejam pretendidas, bem como garantindo a implementação das que são indispensáveis. Estes modelos deram azo à discussão e à convergência do grupo para uma solução unânime, mas também, devido à natureza incremental de complexidade, permitiu distinguir os diferentes componentes e relações entre eles o que resultou numa realização do projeto de acordo com os requisitos estabelecidos. Além da simplificação do problema, a

realização destes modelos permitiu que a criação futura da implementação idealizada fosse feita de forma mais rápida e eficiente.

Na segunda fase, estendeu-se a exploração dos use cases, identificação das classes e criação de diagramas de sequência, entre outros. As principais funcionalidades do projeto foram definidas o que levou a que na terceira fase as decisões tomadas estivessem mais relacionadas com a implementação do sistema e não tanto com o domínio do problema. De facto, o desenvolvimento dos diagramas teve como resultado um desenvolvimento célere da aplicação, eliminando muitos dos problemas que seriam considerados caso não tivessem sido feitos, as vantagens são evidentes. Ainda assim, à medida que avançamos na implementação do projeto, estes diagramas começaram a ficar desatualizados com o aparecimento de casos que não foram considerados e comportamentos que não foram previstos. Com isto dito, somos da opinião que o desenvolvimento destes diagramas, enquanto vantajosa, na medida em que facilita a ideia geral do problema a ser criada, existem problemas que apenas aparecem no momento da implementação, problemas que implicam uma reformulação dos diagramas, reformulação essa que implica o uso de tempo. Esta manutenção dos diagramas revelou ser algo bastante trabalhoso e a principal crítica que temos a fazer ao seu uso.

Passando agora para uma análise crítica da implementação feita, há alguns problemas que temos consciência que existem. A informação das prateleiras está presente em duplicado na classe Mapa: no mapa das prateleiras, e no mapa com as prateleiras de cada corredor com armazenamento. Tínhamos uma versão em que corrigimos este problema, no entanto, como esta não era uma versão estável e com a falta de tempo, acabou por não ser implementado na versão final. A forma como foi feito o DAO das Localizações não foi o melhor. Este problema adveio do facto da chave primária desta tabela ter de ser única, o que era impossível com a implementação que tínhamos desse mapa. Isto introduziu a implicação que no nosso Mapa físico do armazém apenas podem existir 10 prateleiras por corredor com armazenamento. Ainda assim, apesar da solução encontrada não ter sido a mais eficiente ou desejada, continua uma solução funcional.

Conclusão

A construção da 3º fase foi algo consideravelmente difícil, uma vez que, devido à segunda fase estar repleta de diversas inconsistências. Isto obrigou-nos a repensar cada detalhe dos métodos decididos na segunda fase assim como uma reconstrução total do diagrama de classes.

Contudo ao longo do desenvolvimento conseguimos perceber que o planeamento e modelação de uma aplicação é tão ou mais importante que a implementação concreta com a escrita do código, uma vez que dá uma maior possibilidade de organização e planeamento antes do desenvolvimento concreto da aplicação.

Inicialmente, modelando o diagrama de domínio fomos capazes de traduzir o problema que nos foi dado, numa segunda análise, a construção do diagrama de use cases facilitou a escolha e desenvolvimento das funcionalidades para cada tipo de utilizador, por fim a

construção do diagrama de classes serviu como um mapa do sistema em si e os diagramas de sequência ajudaram a refletir como implementar cada use case. No entanto, depois de avançarmos na implementação concreta da aplicação a manutenção destes diagramas passou a ser algo trabalhoso uma vez que o aparecimento de bugs e de comportamentos que não foram previstos eram maiores do que as alterações que poderíamos efetuar.

No geral, podemos dizer que os use cases/requisitos principais que nos foram propostos foram alcançados com sucesso, e que apesar da segunda fase do projeto ter algumas inconsistências, achamos que esta terceira fase atinge muito bem os seus objetivos.