

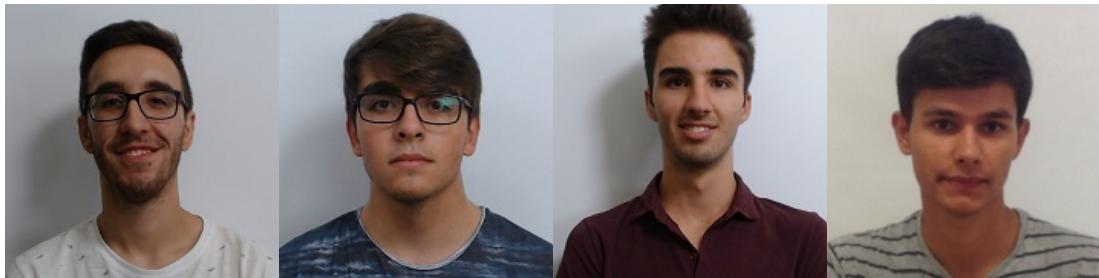
Universidade do Minho

12 de janeiro de 2021

Desenvolvimento de Sistemas de Software

Relatório de grupo 29 - Fase 3

Mestrado Integrado em Engenharia Informática - 3º ano



Adriano Novo
Soto Maior
A89483

Bruno Pinto
Jácome
A89515

José Pedro
Ribeiro Peixoto
A89602

José Luís
Abreu Mendes
A75481

Índice

I. Análise de Requisitos	3
1. Introdução	4
2. Funcionamento do Armazém	5
2.1. Modelo de Domínio	5
2.2. Zonas de armazenamento	5
2.3. Tipos de paletes	5
2.4. Descargas	6
2.5. Requisições	6
2.6. Robots	6
3. Use Case	7
3.1. Diagrama de Use Case	7
3.2. Ator: Leitor	7
3.2.1. Use Case: Registrar Paleta	7
3.3. Ator: Servidor da Produção	8
3.3.1. Use Case: Efetuar Requisição	8
3.4. Ator: Robot	9
3.4.1. Use Case: Notificar robot para transportar (descargas)	9
3.4.2. Use Case: Notificar Robot para transportar (entregas)	10
3.4.3. Use Case: Informar Recolha	10
3.4.4. Use Case: Informar Entrega	11
3.5. Ator: Gestor	11
3.5.1. Use Case: Obter Listagem de Paletes	11
3.5.2. Use Case: Confirmar Pedidos de Descarga	12
3.6. Ator: Motorista	12
3.6.1. Use Case: Efetuar Pedido de Descarga	12
3.6.2. Use Case: Consultar Pedido de Descarga	13
3.7. Ator: Utilizador não autenticado	13
3.7.1. Use Case: Autenticar Utilizador	13
3.8. Ator: Encarregado	14
3.8.1. Use Case: Notificar Satisfação da Requisição	14
4. Conclusão	15
II. Modelação Conceptual da Solução	16
5. Introdução	17
6. Alterações no Modelo de Domínio	18
6.1. Rotas	18
7. Alterações nos Use Cases	19
7.1. Problemas encontrados	19

7.2.	Autor: Utilizador Autenticado	19
7.2.1.	Use Case: Logout	19
7.3.	Diagrama de Use Cases atualizado	20
8.	Lógica de Negócios	21
8.1.	Autor: Leitor	21
8.1.1.	Use Case: Registar Palete	21
8.2.	Autor: Utilizador Não Autenticado	21
8.2.1.	Use Case: Iniciar Sessão	21
8.3.	Autor: Servidor da Produção	21
8.3.1.	Use Case: Efetuar Requisição	21
8.4.	Autor: Robot	22
8.4.1.	Use Case: Notificar Robot para transportar (entregas)	22
8.4.2.	Use Case: Notificar Robot para transportar (descargas)	22
8.4.3.	Use Case: Informar Entrega	22
8.4.4.	Use Case: Informar Recolha	22
8.5.	Autor: Utilizador Autenticado	22
8.5.1.	Use Case: Terminar sessão	22
9.	Diagrama de Componentes	23
9.1.	Subsistemas	23
9.1.1.	SSArmazenamento	23
9.1.2.	SSControloRobot	23
9.1.3.	SSRequisicao	23
9.1.4.	SSUtilizador	23
10.	Diagrama de Classes	24
10.1.	SSArmazenamento	24
10.2.	SSControloRobot	24
10.3.	SSRequisicao	25
10.4.	SSUtilizador	26
11.	Diagramas de Sequência	27
11.1.	addMateriaPrima	27
11.2.	addPaleteDescarga	27
11.3.	addPaleteZonaD	28
11.4.	adicionaAguardarTransporte	28
11.5.	adicionaPaleteEntregar	29
11.6.	adicionaPaleteRequisit	29
11.7.	calculaPaletesExistentes	30
11.8.	calculaRota	30
11.9.	cancelaPedidoEmFalta	30
11.10.	cancelaPedidoTotal	31
11.11.	cancelaRequisicaoEmFalta	31
11.12.	cancelaRequisicaoTotal	31
11.13.	criarPaleteFromQRCode	32
11.14.	informarEntrega	32
11.15.	informarRecolha	33
11.16.	getLocais	33
11.17.	getPraeteiraLivre	33

11.18.getRequisicao	34
11.19.getRota	35
11.20.getRotaCompleta	36
11.21.getZonaDescarga	36
11.22.libertarPaletesReservadas	37
11.23.libertarReservadoPalete	37
11.24.listagemPaletes	38
11.25.login	39
11.26.logout	39
11.27.materiaEqual	39
11.28.mudaEstadoRobot	40
11.29.mudaPaleteRobot	40
11.30.mudaPaletePrateleira	40
11.31.paletesPorLocal	41
11.32.prateleirasLivresPorLocal	42
11.33.procuraMPrima	42
11.34.procuraPalete	43
11.35.procuraPrateleiraLivre	43
11.36.procuraRobot	44
11.37.quantTotalMateriaPrima	44
11.38.registarMateriasPrimasAE Entregar	45
11.39.registarPalete	45
11.40.registarPedido	46
11.41.registarPedidoEmFalta	46
11.42.removeAguardarTransporte	47
11.43.removePaleteEntregar	47
11.44.removePaleteRequisit	47
11.45.removeRequisicao	48
11.46.requisitarPaletesReservadas	48
11.47.reservarMateriaPrima	49
11.48.satisfazerPedido	50
11.49.temEspaco	50
11.50.tentarReservar	51
11.51.tentarReservarPalete	51
11.52.validaInfoPalete	52
11.53.validaMateriaPrima	53
11.54.validaTipoMateriaPrima	53
11.55.validaUtilizador	54
12. Conclusão	55
III. Implementação da Solução	56
13. Introdução	57
14. Correções na Lógica de Negócios	58
14.1. Ator: Gestor	58
14.1.1. Use Case: Obter listagem de Paletes	58

14.2. Ator: Leitor	58
14.2.1. Use Case: Registrar Palete	58
14.3. Ator: Robot	58
14.3.1. Use Case: Notificar Robot para Transportar (descargas)	58
14.3.2. Use Case: Notificar Recolha	58
14.3.3. Use Case: Notificar Entrega	58
15. Novo Diagrama de Classes	59
15.1. Abordagem	59
15.2. SSArmazenamentoFacade	59
15.2.1. Diagrama de Classes	59
15.3. QRCode	59
15.3.1. Diagrama de Classes	59
15.4. SSControloRobotFacade	60
15.4.1. Diagrama de Classes	60
15.5. IGestor	60
15.5.1. Diagrama de Classes	60
16. Base de Dados	61
16.1. Modelo Lógico	61
16.1.1. Diagrama	61
17. Análise Global ao Projeto	62
17.1. Fase 1 : Análise de Requisitos	62
17.2. Fase 2 : Modelação Conceptual da Solução	62
17.3. Fase 3 : Implementação da Solução	62
18. Conclusão	63

Fase I.

Análise de Requisitos

1 | Introdução

Este relatório foi realizado no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, envolvendo o conceito de Modelos de Domínio e Use Cases. Tem como objetivo principal ilustrar o comportamento do Sistema de Gestão de Stocks de um armazém de paletes - que irá ser abreviado apenas para Sistema - e explicar o seu funcionamento e o seu fundamento. O relatório divide-se essencialmente em 3 partes:

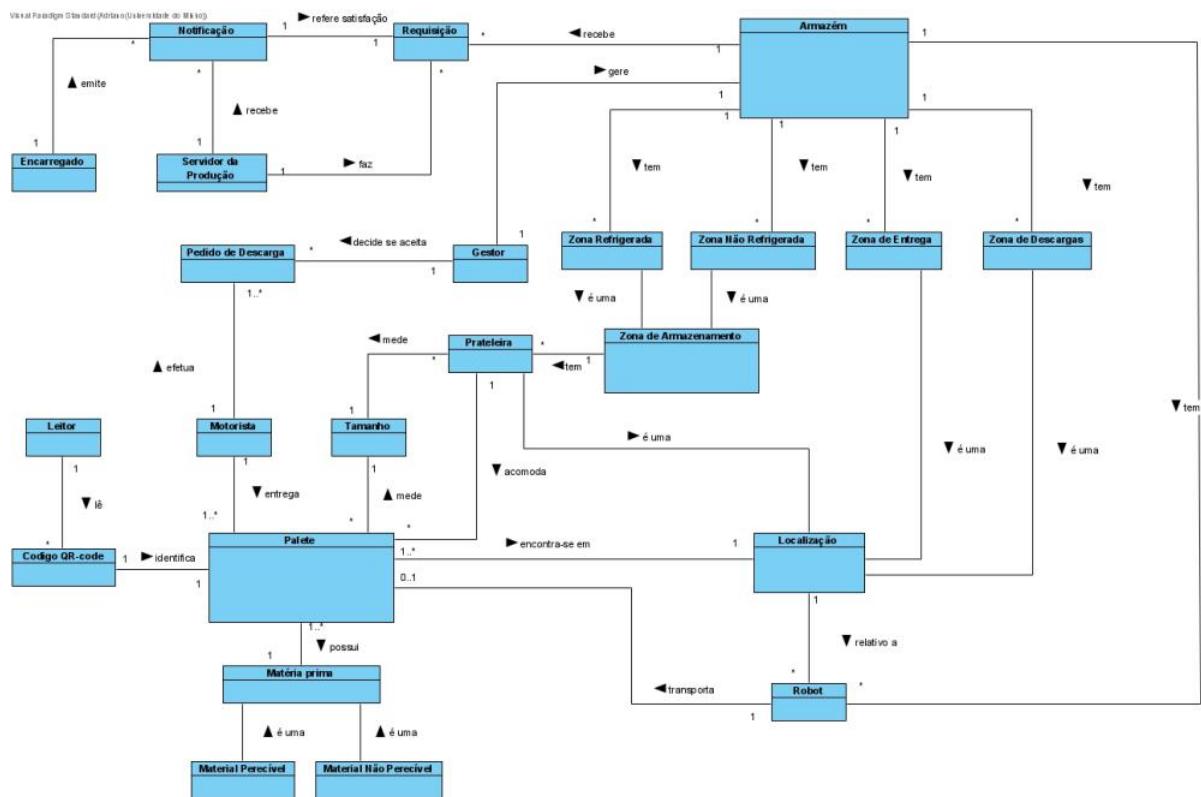
Na primeira parte, Funcionamento do Armazém, relata-se o dinamismo envolvendo as suas diferentes entidades dentro do armazém.

Já a segunda parte, Use Case, é bastante longa já que pretende descrever todos os passos de cada Use Case do Sistema, assim como, a abordagem à tarefa associada.

Em último lugar, concluímos o relatório com algumas considerações gerais acerca do trabalho realizado.

2 | Funcionamento do Armazém

2.1. Modelo de Domínio



2.2. Zonas de armazenamento

Este armazém de paletes está dividido em várias zonas: zona refrigerada, zona não refrigerada, zona de descarga e zona de entrega.

Enquanto que o armazém apenas tem uma só zona de descarga e zona de entrega, este poderá ter várias zonas refrigeradas ou não refrigeradas. Contudo terá sempre pelo menos um de cada tipo de zona. Além disso, tanto a zona refrigerada como a zona não refrigerada são uma zona de armazenamento, isto é, uma zona com prateleiras para armazenar paletes.

2.3. Tipos de paletes

A distinção dos dois tipos de zona anteriores (refrigerada e não refrigerada) são importantes, porque as paletes em questão têm tamanhos variados e são constituídas por matéria prima de dois tipos: perecível e não perecível.

2.4. Descargas

Os pedidos de descargas no armazém são criados e enviados por motoristas à entrada do armazém e o pedido é, posteriormente, analisado pelo gestor que poderá autorizar esta descarga se a capacidade do armazém assim o permitir.

Quando um pedido de descarga é autorizado, regista-se as paletes no sistema, através de um leitor de QR-code e só posteriormente é que são armazenadas.

2.5. Requisições

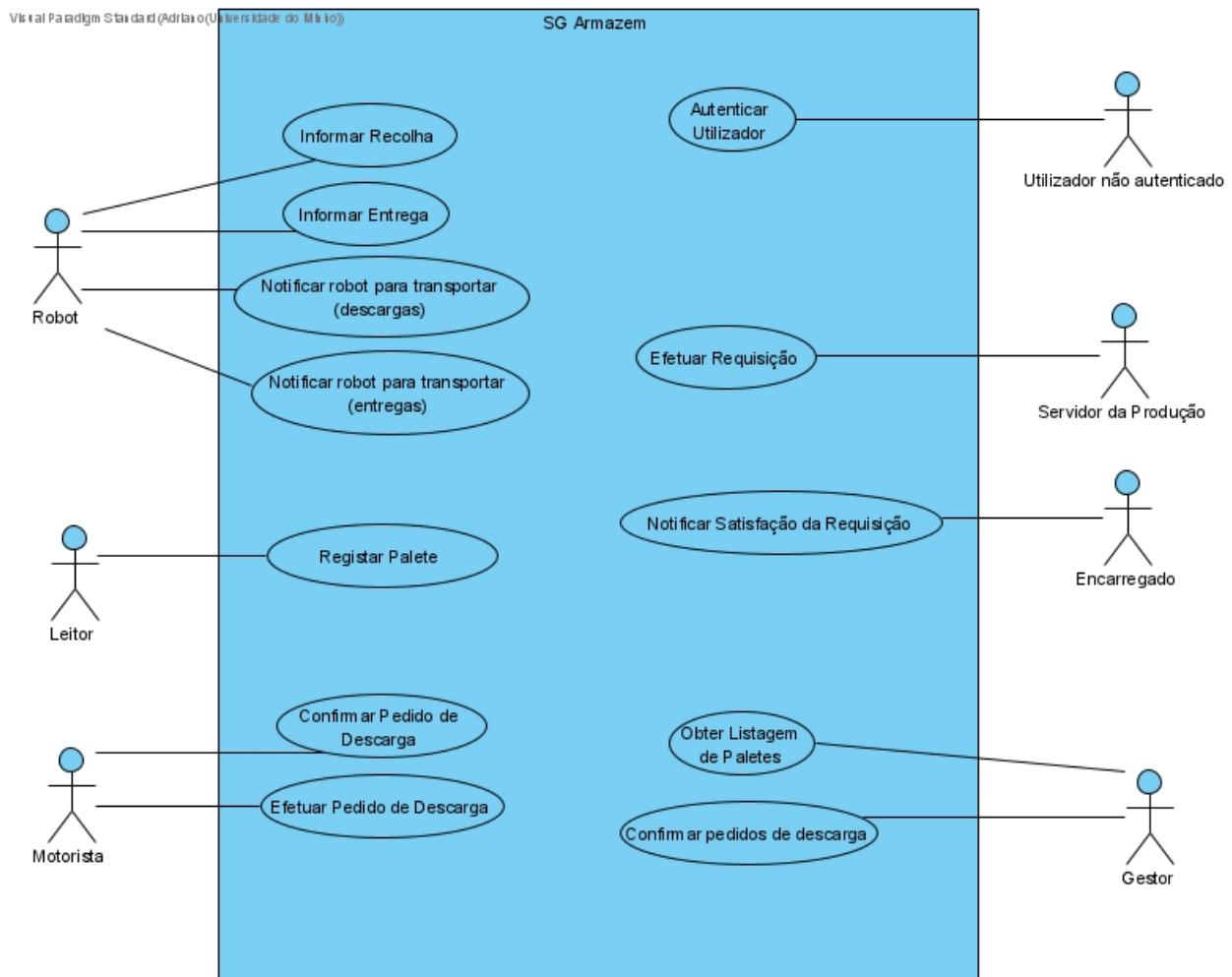
Já as requisições de paletes são feitas por um sistema externo, o Servidor da Produção. O sistema verificará se consegue responder às necessidades do pedido e em caso afirmativo o processo de transporte das paletes para a zona de entrega começa. No fim, o Encarregado do armazém notifica o Servidor que a requisição foi satisfeita.

2.6. Robots

O transporte de paletes dentro do armazém é feito via Robot. O Sistema indica-lhes a paleta a transportar e a sua localização, bem como o trajeto a percorrer. Estes robots sinalizam o sistema sempre que recolhem ou entregam uma paleta.

3 | Use Case

3.1. Diagrama de Use Case



3.2. Ator: Leitor

3.2.1. Use Case: Registrar Paleta

Cenário: Leitor regista uma paleta no sistema.

Pré-Condição: Há, pelo menos, uma paleta não registada na zona de receção.

Pós-condição: A paleta fica registada no sistema de gestão de stocks.

Fluxo normal

1. Leitor comunica com o sistema as informações do *QR-code* sobre a paleta.
2. Sistema valida todas as informações sobre a paleta.
3. Sistema determina o tipo de matéria-prima da paleta.

4. Sistema fica com um registo da palete.
5. Sistema adiciona a palete à lista de paletes a entregar.
6. Sistema sinaliza o Leitor que a palete foi registada.

Fluxo de Exceção 1: [Código de QR-code não é válido] (passo 2)

- 2.1 Sistema sinaliza o leitor que as informações não são válidas.

Abordagem

Uma máquina, que designamos por leitor, após fazer a leitura de um QR-code, correspondente a uma paleta, comunicará com o sistema de gestão de stocks, passando-lhe informações sobre as paletes (como a sua dimensão e o tipo de material). O Sistema tem de confirmar que recebeu todas as informações necessárias, caso contrário não poderá aceitar o registo da paleta, porque não saberia como procurar paletes por determinadas características. O sistema vai adicionar essa paleta na lista de paletes a entregar, pois já é possível um robot levar essa paleta para a prateleira. Caso o sistema não tenha recebido todas as informações necessárias para identificar uma paleta, não é possível fazer o seu registo e o leitor é avisado que a funcionalidade não foi concluída com sucesso.

3.3. Ator: Servidor da Produção

3.3.1. Use Case: Efetuar Requisição

Cenário: Servidor da produção transmite ao sistema uma requisição de paletes.

Pré-Condição: True.

Pós-condição: O Sistema fica com o registo das paletes que foram requisitadas.

Fluxo normal

1. Servidor da Produção comunica ao sistema as paletes necessárias do pedido.
2. Sistema valida a disponibilidade das paletes n
3. Sistema cria registo das paletes a entregar.

Fluxo alternativo 1: [Alguma paleta não disponível] (passo 2)

- 2.1 Sistema comunica ao Servidor da Produção que as paletes não estão disponíveis de momento.
- 2.2 Servidor da Produção pede o cancelamento das paletes em falta.

Fluxo alternativo 2: [pedido por fases] (passo 2.2)

- 2.2.1 Servidor da Produção confirma pedido total.
- 2.2.2 Servidor da Produção cria registo de paletes em falta para entrega posterior.

Fluxo de Exceção 1: [pedido não é parcial] (passo 2.2)

- 2.2.1 Servidor da Produção cancela pedido.

Abordagem

Caso algum cliente a que chamamos servidor de produção precise de material que esteja no armazém, é necessário que seja feito ao sistema uma requisição de paletes, identificando as suas características. O Sistema vai verificar se o armazém consegue satisfazer as exigências da requisição. Se sim, então podemos criar um registo que peça que sejam entregues todas essas paletes. Caso contrário, o Servidor de Produção poderá cancelar o pedido e não será entregue nenhuma paleta, ou então, poderá receber apenas as paletes disponíveis nas prateleiras do armazém (pedido parcial). Este ainda pode receber as paletes pedidas existentes nas prateleiras do armazém e receber as restantes paletes posteriormente. Quando houver stock será feito um registo de paletes a entregar constituído pelas paletes em falta.

3.4. Ator: Robot

3.4.1. Use Case: Notificar robot para transportar (descargas)

Cenário: Foram descarregadas paletes que necessitam de transporte para prateleiras.

Pré-Condição: Fila de paletes a entregar não está vazia.

Pós-condição: O robot fica notificado e palete foi registada como estando a aguardar transporte na zona de descarga.

Fluxo normal

1. Sistema retira palete da fila de paletes a entregar.
2. Sistema procura prateleira para colocar palete.
3. Sistema procura robot para transportar palete.
4. Sistema calcula rotas para o robot.
5. Sistema comunica rotas e palete ao robot.
6. Sistema altera o estado do robot para ocupado.
7. Sistema regista a paleta na lista de paletes a aguardar transporte.

Fluxo de Exceção 1: [Não há robots disponíveis] (passo 3)

- 3.1 Sistema volta a inserir a paleta na fila de paletes a entregar.

Abordagem

De modo a armazenar as paletes recebidas, o sistema deve escolher robots para o seu transporte. Neste processo, o sistema remove uma paleta da lista de paletes a entregar, procura uma prateleira adequada e seleciona um robot para o seu transporte, calculando e informando-o do trajeto a seguir, alterando o seu estado para ‘ocupado’. A paleta é, então, transferida para a lista de paletes a aguardar transporte. Este Use Case gera um Fluxo de Exceção quando não há robots livres.

3.4.2. Use Case: Notificar Robot para transportar (entregas)

Cenário: Foi feito, pelo menos, um pedido de entrega, por isso, é preciso transportar paletes para zona de entregas.

Pré-Condição: Fila de paletes requisitadas não está vazia.

Pós-condição: O robot fica notificado e palete foi registada como estando a aguardar transporte na prateleira.

Fluxo normal

1. Sistema retira palete da fila de paletes requisitadas.
2. Sistema procura robot para transportar palete.
3. Sistema calcula rotas para o robot.
4. Sistema comunica rotas e palete ao robot.
5. Sistema altera o estado do robot para ocupado.
6. Sistema regista a paleta na lista de paletes a aguardar transporte.

Fluxo de Exceção 1: [Não há robots disponíveis] (passo 2)

- 2.1 Sistema volta a inserir a paleta na fila de paletes requisitadas.

Abordagem

Este Use Case assemelha-se ao anterior (3.4.1), sendo que a única diferença é que as paletes são retiradas da fila de paletes requisitadas.

3.4.3. Use Case: Informar Recolha

Cenário: Robot recolhe paleta.

Pré-Condição: O robot já recebeu a sua rota e levantou a paleta.

Pós-condição: O sistema é informado da recolha da paleta.

Fluxo normal

1. Robot informa o sistema da recolha da paleta.
2. Sistema remove a paleta da lista de paletes a aguardar transporte.
3. Sistema atualiza localização da paleta.

Abordagem

Para manter o sistema a par do estado do transporte da paleta, o robot envia um sinal quando recolhe a paleta. Após a receção do sinal, o sistema remove a paleta da fila de paletes a aguardar transporte e atualiza a sua localização e o robot está agora disponível.

Este processo garante que o sistema se mantenha sempre o mais atualizado possível, permitindo uma melhor compreensão do estado atual do armazém.

3.4.4. Use Case: Informar Entrega

Cenário: Robot entrega palete.

Pré-Condição: O robot já completou o seu percurso e entregou a palete.

Pós-condição: O sistema é informado da entrega da palete.

Fluxo normal

1. Robot informa o sistema da entrega da palete.
2. Sistema atualiza a localização da palete.

Abordagem

Após a entrega da paleta na prateleira, o robot informa o sistema da conclusão da sua tarefa. O sistema atualiza uma última vez a localização da paleta, assim como o estado do robot, de ocupado para livre, permitindo que este aceite novos pedidos de transporte.

3.5. Ator: Gestor

3.5.1. Use Case: Obter Listagem de Paletes

Cenário: O gestor pede uma listagem com a localização das paletes existentes no armazém.

Pré-Condição: O Gestor está autenticado.

Pós-condição: O Gestor tem conhecimento da listagem.

Fluxo normal

1. Gestor pede listagem ao sistema.
2. Sistema verifica paletes existentes.
3. Sistema agrupa paletes por zona.
4. Sistema calcula o número de prateleiras livres em cada zona.
5. Sistema mostra ao gestor a listagem de paletes.

Fluxo Alternativo 1: [Não há paletes] (passo 2)

- 2.1 Sistema informa ao Gestor que as prateleiras estão todas livres.

Abordagem

Neste use case, o gestor limita-se a pedir uma listagem de paletes ao Sistema. Depois, o Sistema, de modo a poder enviar esta listagem, verifica as paletes que se encontram no armazém no momento, agrupa as paletes por zona, calcula os espaços livres em cada zona e, por fim, mostra ao gestor esta informação.

Além disso, não há nenhum fluxo de exceção uma vez que o único cenário onde poderia haver problemas seria ao verificar a existência de paletes. Neste caso, o gestor fica com conhecimento da listagem de paletes apesar de ser nula, verificando assim a pós-condição.

3.5.2. Use Case: Confirmar Pedidos de Descarga

Cenário: Há um motorista à entrada do armazém à espera de uma resposta ao seu pedido de descarga.

Pré-Condição: Existem registo de pedidos de descarga na lista.

Pós-condição: O registo do pedido de descarga sai da lista.

Fluxo normal

1. Sistema mostra ao gestor a lista de pedidos de descarga e o nível de ocupação.
2. Gestor aceita um pedido de descarga.
3. Sistema adiciona pedido de descarga à lista de registo de pedidos, com estado de aceite.
4. Sistema remove o pedido de descarga da lista.

Fluxo Alternativo 1: [Gestor rejeita pedido] (passo 2)

- 2.1 Gestor rejeita, justificando através de um curto comentário.
- 2.2 Sistema adiciona pedido dedescarga à lista de registo de pedidos, comestado de rejeitado.
- 2.3 Sistema remove pedido dedescarga da lista de espera.

Abordagem

Neste use case, o Sistema mostra ao gestor a lista de pedidos de descarga juntamente com o nível de ocupação. O Gestor, baseado nas informações obtidas, pode aceitar ou não o pedido, sendo este removido da lista de espera de pedidos de descarga. Além disso, é importante notar que o motorista e o ator estão relacionados indiretamente. Isto não permitiu que certas tarefas fossem realizadas num só use case.

3.6. Ator: Motorista

3.6.1. Use Case: Efetuar Pedido de Descarga

Cenário: O Motorista faz um pedido de descarga ao Sistema.

Pré-Condição: O motorista está à entrada do armazém.

Pós-condição: O Sistema fica com o registo do pedido de descarga.

Fluxo normal

1. Motorista cria um pedido de descarga.
2. Motorista preenche pedido indicando quantas paletes precisam de refrigeração, e seus tamanhos.
3. Motorista preenche pedido indicando quantas paletes não precisam de refrigeração, e seus tamanhos.
4. Motorista completa pedido e envia-o ao sistema.
5. Sistema adiciona o pedido à lista de espera de pedidos de descarga.
6. Sistema sinaliza ao motorista que o pedido foi registado.

Abordagem

Para efetuar pedidos de descarga, o Motorista entra em contacto com o Sistema para criar e enviar um pedido. Neste processo, o motorista descreve as quantidades e qualidades das paletes que transporta. O pedido criado é enviado para o sistema, registando-o numa lista de espera de pedidos de descarga.

Além disso, como aconteceu em 3.5.2, foi necessário dividir a tarefa de "descarregar" em dois use cases: Use Case: **Efetuar Pedido de Descarga** e Use Case: **Consultar Pedido de Descarga**. Como não é o Sistema a analisar e decidir sobre um pedido de descagar, este age como um intermediário entre dois atores distintos, o motorista e o gestor. Caso fosse o Sistema a gerir o processo de aceitar ou rejeitar estes pedidos, poderiam surgir fluxos alternativos com descargas parciais se o armazém estivesse parcialmente lotado, que poderia ser aceite ou rejeitada pela motorista. No caso de rejeição, teríamos um fluxo de exceção em que as paletes não eram entregues ao armazém e o pedido de descarga era cancelado.

3.6.2. Use Case: Consultar Pedido de Descarga

Cenário: O Motorista consulta o pedido de descarga ao Sistema.

Pré-Condição: O motorista está à entrada do armazém.

Pós-condição: O motorista está autorizado a descarregar.

Fluxo normal

1. Motorista consulta pedido de descarga.
2. Sistema informa que o pedido foi aceite e que está autorizado a descarregar.

Fluxo de Exceção 1: [Pedido foi rejeitado] (passo 2)

- 2.1 Sistema informa que o pedido foi rejeitado e apresenta um curto comentário.
- 2.2 Sistema não autoriza motorista a descarregar.

Fluxo de Exceção 2: [Pedido ainda não foi avaliado] (passo 2)

- 2.1 Sistema informa que o pedido ainda está à espera de ser avaliado.

Abordagem

Neste use case, o Motorista tem a possibilidade de consultar o estado do seu pedido, sendo que se for aceite ele é autorizado a descarregar.

Este use case surge para ser possível ao Sistema comunicar ao motorista o estado do seu pedido. Como foi discutido anteriormente em 3.6.1, se o Sistema decidisse automaticamente se a descarga era possível este use case seria desnecessário. Portanto, este é um método de concluir a interação entre o Sistema e o Motorista.

3.7. Ator: Utilizador não autenticado

3.7.1. Use Case: Autenticar Utilizador

Cenário: Há um utilizador que pretende se autenticar no sistema.

Pré-Condição: O utilizador já está registado.

Pós-condição: O utilizador passa ter permissões de Gestor ou Encarregado.

Fluxo normal

1. Cliente apresenta o seu nome de utilizador e palavra-passe.
2. Sistema valida os dados.
3. Sistema comunica ao utilizador que entrou no sistema como Gestor ou Encarregado.

Fluxo de Exceção 1: [Dados inválidos] (passo 2)

- 2.1 Sistema informa o Utilizador do erro nos dados.

Abordagem

A autenticação tem como objetivo identificar um utilizador de modo a restringir o acesso a partes do sistema. Esta autenticação é feita através de um identificador único de cada pessoa que interaja com o sistema.

3.8. Ator: Encarregado

3.8.1. Use Case: Notificar Satisfação da Requisição

Cenário: O Encarregado notifica a satisfação da requisição.

Pré-Condição: A requisição existe.

Pós-condição: A requisição é registada como completa no Sistema.

Fluxo normal

1. Encarregado pede ao Sistema a listagem das requisições.
2. Sistema calcula o estado das requisições.
3. Sistema devolve ao Encarregado numa lista com os estados das requisições.
4. Encarregado comunica ao sistema as requisições que pretende notificar a satisfação.
5. Sistema atualiza as requisições por notificar.
6. Sistema informa o encarregado que as requisições escolhidas foram satisfeitas.

Abordagem

A principal função do encarregado é notificar a satisfação das requisições. Para isso, este pede ao sistema a listagem das requisições, acompanhadas pelo seu estado. Além disso, ele deve notificar a aquelas que já foram concluídas, como satisfeitas ou insatisffeitas.

4 | Conclusão

O grupo considerou que esta primeira fase do projeto foi simples, embora trabalhosa, uma vez que o modelo de domínio e os Use Case servem mais de preliminares, não representando uma componente programável do Sistema de Gestão de Stocks.

Ainda assim, o grupo teve, inicialmente, algumas dificuldades de organizar as entidades envolventes no armazém, por exemplo a inexistência de um cliente concreto gerou dúvida já que impossibilitava uma comunicação, quer para o questionar sobre decisões, quer para saber que funcionalidades a implementar.

No entanto, ao resolver conflitos ou impossibilidades da comunicação do sistema com os atores, o grupo acabou por ficar mais esclarecido. Por exemplo, achamos necessário criar-se um ator "Utilizador Não Autenticado" para tornar a autenticação um método possível para o Gestor e Encarregado.

No fim, achamos que o modelo de domínio e os use cases seguem os requisitos do enunciado, possibilitando a continuação do projeto para a próxima fase.

Fase II.

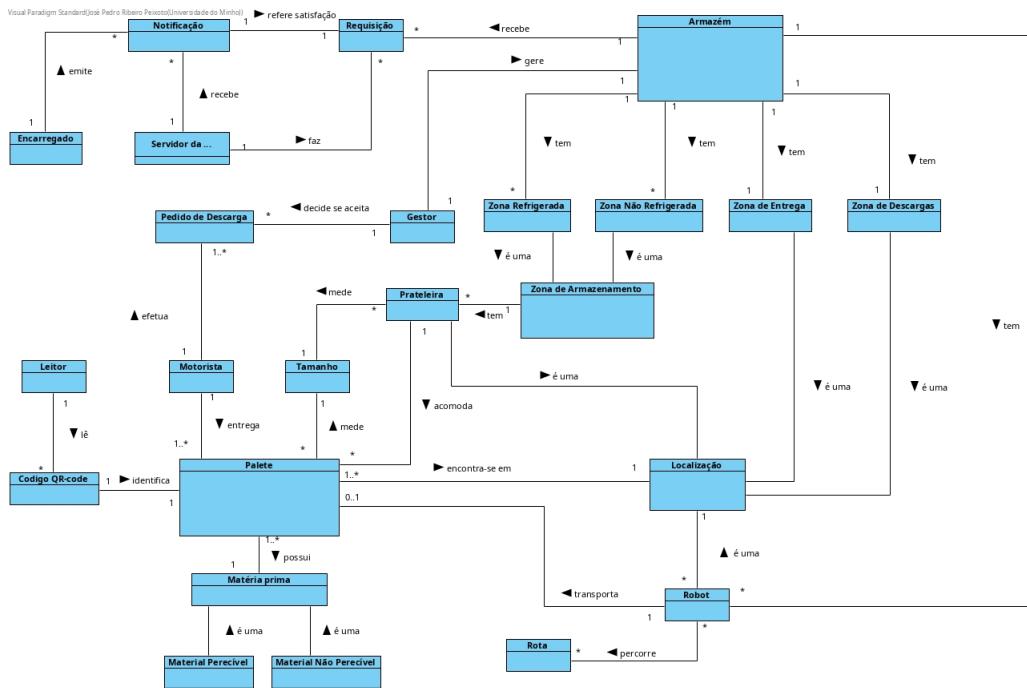
Modelação Conceptual da Solução

5 | Introdução

Este relatório foi realizado no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, envolvendo o conceito de Modelos de Domínio e Use Cases. Tem como objetivo principal ilustrar o comportamento do Sistema de Gestão de Stocks de um armazém de paletes - que irá ser abreviado apenas para Sistema - e explicar o seu funcionamento e o seu fundamento.

O relatório divide-se essencialmente em 6 partes: Na primeira parte, apresentam-se as alterações desde a realização da fase anterior. Já na segunda parte, indentificam-se as transações da lógica de negócios através de tabelas. Depois, mostra-se o diagrama de componentes e a abordagem adotada pelo grupo. A seguir, apresenta-se os diagramas de classes dos diversos subsistemas. Em penúltimo lugar, ilustra-se uma longíssima lista de diagramas de sequência. Para concluir, são referidas algumas considerações gerais acerca do trabalho realizado.

6 | Alterações no Modelo de Domínio



6.1. Rotas

Uma vez que o sistema terá de informar os Robots de rotas (para transporte de paletes), foi necessário adicionar ao modelo de domínio a entidade Rota.

7 | Alterações nos Use Cases

7.1. Problemas encontrados

Logout de um utilizador autenticado

Ao analisar o diagrama de Use Case da Fase I, verifica-se que não existe nenhum método para que o Gestor e o Encarregado possa terminar sessão.

Utilizador Autenticado

Além disso, notemos que tanto o Gestor como o Encarregado podem terminar sessão. Sendo assim, achou-se mais elegante criar o ator **Utilizador Autenticado** para englobar os use cases que possam haver em comum (que neste caso é apenas 1). A razão é idêntica à da existência do ator Utilizador Não Autenticado

7.2. Ator: Utilizador Autenticado

7.2.1. Use Case: Logout

Cenário: Há um utilizador que pretende terminar sessão.

Pré-Condição: O utilizador já está autenticado.

Pós-condição: O utilizador deixa de ter acesso ao Sistema.

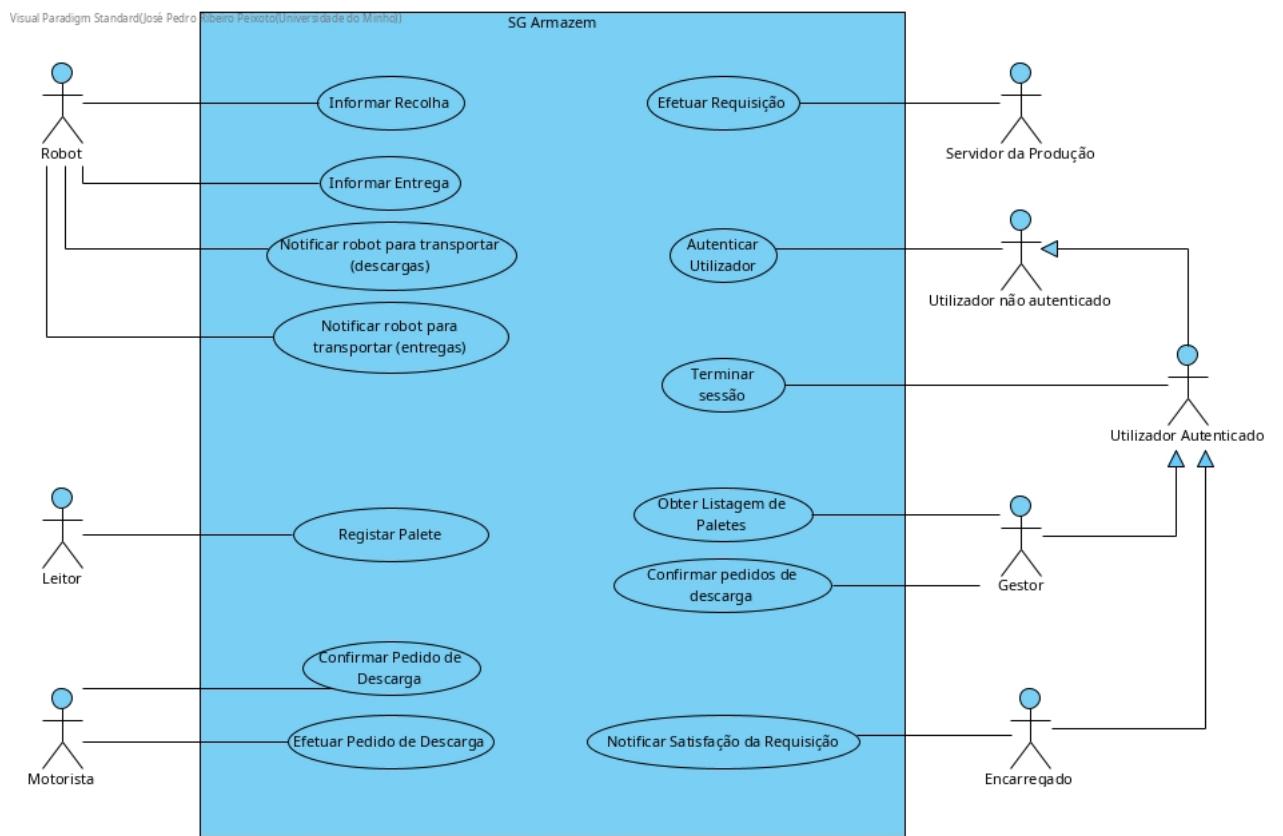
Fluxo normal

1. Autor seleciona a opção de terminar sessão.
2. Sistema confirma a saída do utilizador

Abordagem

O Ator limita-se a desconectar-se do Sistema.

7.3. Diagrama de Use Cases atualizado



8 | Lógica de Negócios

8.1. Ator: Leitor

8.1.1. Use Case: Registar Palete

Use Case: Registar Palete				
Fluxo Normal	1	Leitor comunica com o sistema as informações do QR-code sobre a paleta	verificar se todas as informações sobre a paleta estão correctas	SSArmazenamento
	2	Sistema valida todas as informações sobre a paleta		
	3	Sistema determina o tipo de matéria-prima da paleta		
	4	Sistema fica com um registo da paleta	registar uma paleta no sistema	+registarPaleta(codigoQR : QRCode) : Boolean SSArmazenamento
	5	Sistema adiciona a paleta à lista de paletes a entregar	adicionar paleta registada na zona de descargas	+addPaletaDescarga(idPaleta : String) SSArmazenamento
	6	Sistema sinaliza o Leitor que a paleta foi registada	registar uma paleta no sistema	+registarPaleta(codigoQR : QRCode) : Boolean SSArmazenamento
Fluxo de Exceção 1	2.1	Sistema sinaliza o leitor que as informações não são válidas.	este passo é realizado a partir da obtenção do valor de retorno do método +validaInfoPaleta	-validaInfoPaleta(codigoQR : QRCode) : Boolean SSArmazenamento

8.2. Ator: Utilizador Não Autenticado

8.2.1. Use Case: Iniciar Sessão

Use Case: Autenticar Utilizador				
Fluxo Normal	1	Utilizador não autenticado apresenta o seu nome de utilizador e palavra-passe	Autenticação	+login(idUtilizador : String, password : String) SSUtilizador
	2	Sistema valida os dados	Validação	-validaUtilizador(idUtilizador : String, password : String) : boolean SSUtilizador
	3	Sistema comunica ao utilizador que entrou no sistema como Gestor ou Encarregado	Autenticação	+login(idUtilizador : String, password : String) SSUtilizador
Fluxo de Exceção 1	2.1	Sistema informa o Utilizador do erro nos dados	Validação	-validaUtilizador(idUtilizador : String, password : String) : boolean SSUtilizador

8.3. Ator: Servidor da Produção

8.3.1. Use Case: Efetuar Requisição

Use Case: Efetuar Requisição				
Fluxo Normal	1	Servidor da Produção comunica ao sistema as matérias-primas necessárias do pedido	verifica se as paletes com as matérias-primas do pedido estão disponíveis	+verificarDisponibilidadePedido(pedido : Pedido) : Boolean SSArmazenamento
	2	Sistema valida a disponibilidade das matérias-primas		
	3	Sistema cria registo das matérias-primas a entregar	requisitar as paletes das matérias-primas necessárias	+registrarMateriasPrimasAEntregar(pedido : Pedido) : Boolean SSRequisicao
Fluxo Alternativo 1	2.1	Sistema comunica ao Servidor da Produção que as matérias-primas não estão disponíveis de momento	cancelar a requisição de apenas as matérias-primas que não estão disponíveis neste momento	+cancelaPedidoEmFalta(idReq : String) SSRequisicao
	2.2	Servidor da Produção pede o cancelamento das matérias-primas em falta		
Fluxo Alternativo 2	2.2.1	Servidor da Produção confirma pedido total	registar as matérias-primas em falta como uma requisição para entrega posterior	+registrarPedidoEmFalta(idReq : String) SSRequisicao
	2.2.2	Servidor da Produção cria registo de matérias-primas em falta para entrega posterior		
Fluxo de Exceção 1	2.2.1	Servidor da Produção cancela pedido	cancelar a requisição de todas as matérias primas do pedido	+cancelaPedidoTotal(idReq : String) SSRequisicao

8.4. Ator: Robot

8.4.1. Use Case: Notificar Robot para transportar (entregas)

Use Case: Notificar Robot para transportar (entregas)				
Fluxo Normal	1	Sistema retira palete da fila de paletes requisitadas	remover palete da lista de paletes a requisitadas	+ removePaletaRequisit(idP : Palete) SSArmazenamento
	2	Sistema procura robot para transportar palete	procurar um robot para transportar a palete para a zona de entrega	+ procuraRobot(p : Palete) : Robot SSControloRobot
	3	Sistema calcula rotas para o robot	calcular rota para o robot	+ calculaRota(r : Robot, idLocalPaleta : String, destino : String) : Rota SSControloRobot
	4	Sistema comunica rotas e palete ao robot	procurar um robot para transportar a palete para a zona de entrega	+ procuraRobot(p : Palete) : Robot SSControloRobot
	5	Sistema altera o estado do robot para ocupado	atualizar o estado do robot de livre para ocupado	+ mudaEstadoRobot(r : Robot) : void SSControloRobot
	6	Sistema regista a palete na lista de paletes a aguardar transporte	registar uma palete na lista de paletes a aguardar transporte	+ adicionaAguardarTransporte(idP : String) SSArmazenamento
Fluxo de Exceção 1	2.1	Sistema volta a inserir a palete na fila de paletes requisitadas	adicionar palete na lista de paletes requisitadas	+ adicionaPaletaRequisit(idP : Palete) SSArmazenamento

8.4.2. Use Case: Notificar Robot para transportar (descargas)

Use Case: Notificar Robot para transportar (descargas)				
Fluxo Normal	1	Sistema retira palete da fila de paletes a entregar	remover palete da lista de paletes a entregar	+ removePaletaEntregar(idP : Palete) SSArmazenamento
	2	Sistema procura prateleira para colocar palete	procurar uma prateleira que ainda tenha espaço para uma palete, desde que esta caiba	+ procuraPrateleiraLivre(idPaleta : String) : Prateleira SSArmazenamento
	3	Sistema procura robot para transportar palete	procurar um robot para transportar a palete para a prateleira	+ procuraRobot(p : Palete) : Robot SSControloRobot
	4	Sistema calcula rotas para o robot	procurar um robot para transportar a palete para a zona de entrega	+ calculaRota(r : Robot, idLocalPaleta : String, destino : String) : Rota SSControloRobot
	5	Sistema comunica rotas e palete ao robot	atualizar o estado do robot de livre para ocupado	+ mudaEstadoRobot(r : Robot) : void SSControloRobot
	6	Sistema regista a palete na lista de paletes a aguardar transporte (de robot)	registar uma palete na lista de paletes a aguardar transporte	+ adicionaAguardarTransporte(idPaleta : String) SSArmazenamento
Fluxo de Exceção 1	2.1	Sistema volta a inserir a palete na fila de paletes a entregar	adicionar palete na lista de paletes requisitadas	+ adicionaPaletaEntregar(idP : String) SSArmazenamento

8.4.3. Use Case: Informar Entrega

Use Case: Informar Entrega				
Fluxo Normal	1	Robot informa o sistema da entrega da paleta	Informar que robot entregou a paleta	+ informarEntregar(r : Robot, idPrateleira : String) SSControloRobot
	2	Sistema atualiza a localização da paleta	Alteração do Local da Paleta	+ mudaPaletaPrateleira(prateleira : Prateleira, paleta : Paleta, local : Local, idPrateleira : String) SSArmazenamento
	3	Robot atualiza o seu estado para livre	Alteração do Estado do Robot	+ mudaEstadoRobot(robot : Robot) SSControloRobot

8.4.4. Use Case: Informar Recolha

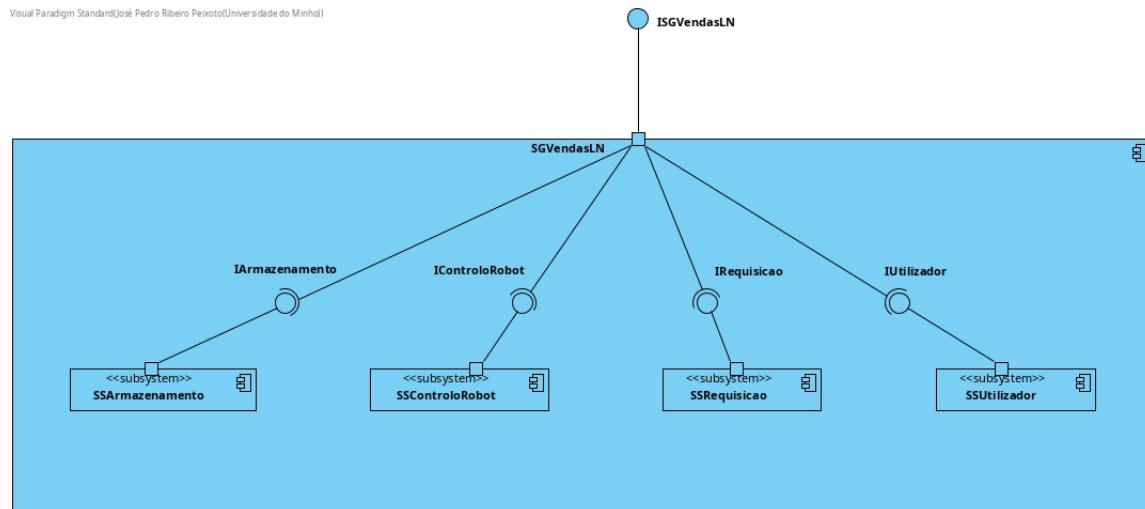
Use Case: Informar Recolha				
Fluxo Normal	1	Robot informa o sistema da recolha da paleta	Informar que robot recolheu a paleta	+ informarRecolha(r : Robot) SSControloRobot
	2	Sistema remove a paleta da lista de paletes a aguardar transporte	Remoção da Paleta da Lista a Aguardar Transporte	+ removeAguardarTransporte(paleta : Paleta) SSArmazenamento
	3	Sistema atualiza localização da paleta	Alteração da Posição da Paleta	+ mudaPaletaRobot(paleta : Paleta) SSArmazenamento

8.5. Ator: Utilizador Autenticado

8.5.1. Use Case: Terminar sessão

Use Case: Terminar Sessão				
Fluxo Normal	1	Utilizador autenticado seleciona a opção de terminar sessão	Terminar Sessão	+ logout() SSUtilizador
	2	Sistema confirma a saída do utilizador		

9 | Diagrama de Componentes



9.1. Subsistemas

Com base na fase anterior e na análise da lógica de negócios apresentada anteriormente, decidiu-se dividir o sistema em 4 subsistemas: **SSArmazenamento**, **SSControloRobot**, **SSRequisicao** e **SSUtilizador**. Embora seja evidente o que cada um destes representa, apresentar-se-á uma breve descrição de cada um deles.

9.1.1. SSArmazenamento

Este subsistema é o coração de todo o projeto. Representa o núcleo e, como tal, a maior parte da carga de trabalho e memória irá estar lá. Isso será visto mais à frente no diagrama de classes.

Muito resumidamente, este subsistema não contém apenas métodos e classes restringidas a um dinamismo intra-armazém, mas também envolverá métodos que estabeleçam comunicação entre este e as múltiplas classes do programa. Por exemplo: requisitar paletes, pedir listagem de paletes, ...

9.1.2. SSControloRobot

Este subsistema trata do controlo de Robots para, por exemplo, dar rotas ou procurar por robots livres para transportarem paletes.

9.1.3. SSRequisicao

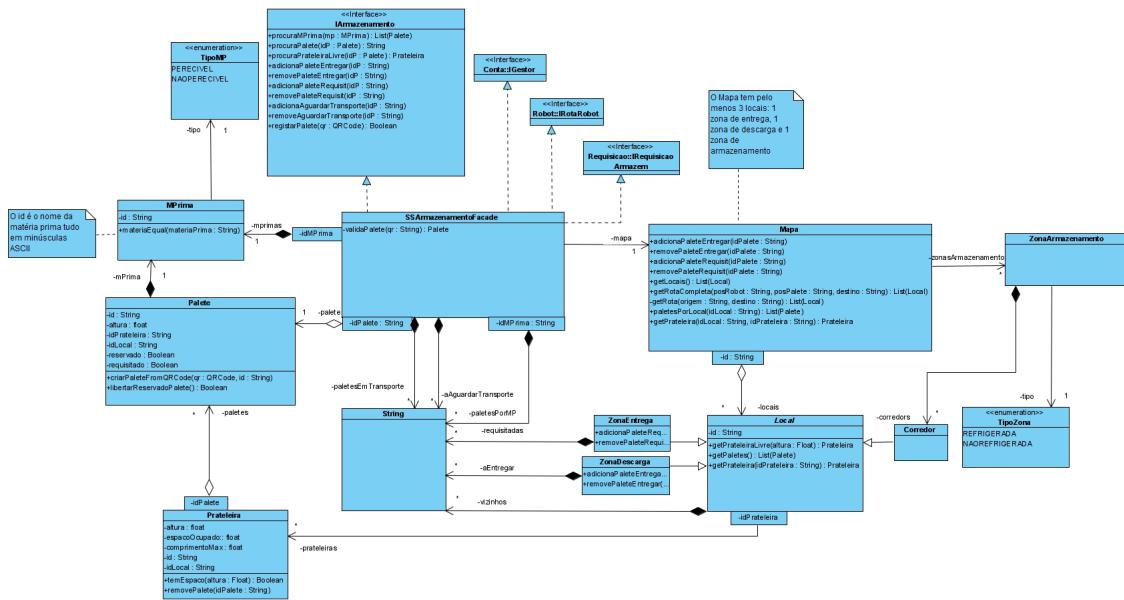
Este subsistema trata do "diálogo" entre o sistema e os pedidos de requisição, bem como o seu registo, de modo a captar todos os problemas (ou não) existentes durante os seus processos.

9.1.4. SSUtilizador

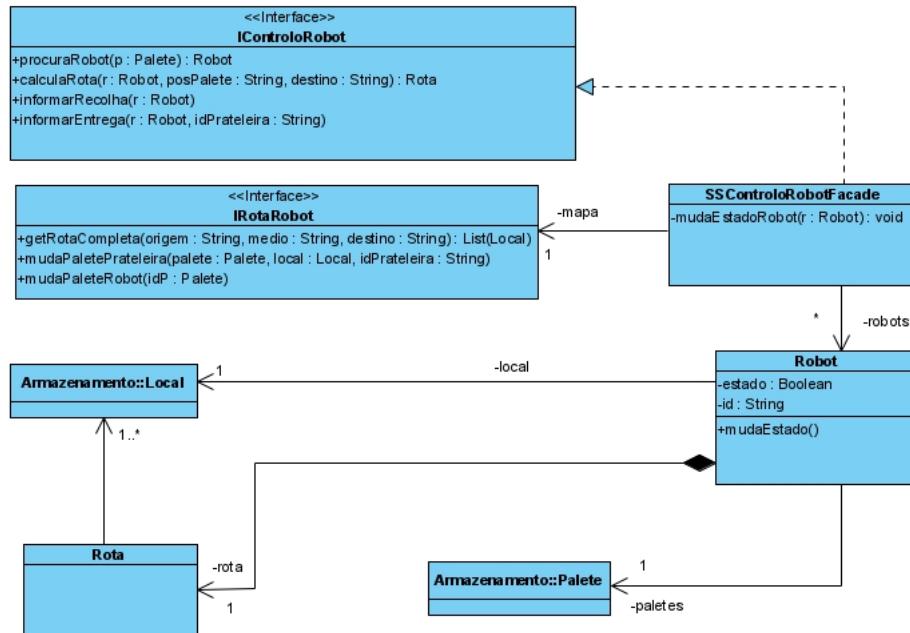
Este subsistema trata da interatividade do sistema com alguns dos atores, privilegiando-se o Gestor que usufrui dos métodos mais importantes de gestão do Armazém.

10 | Diagrama de Classes

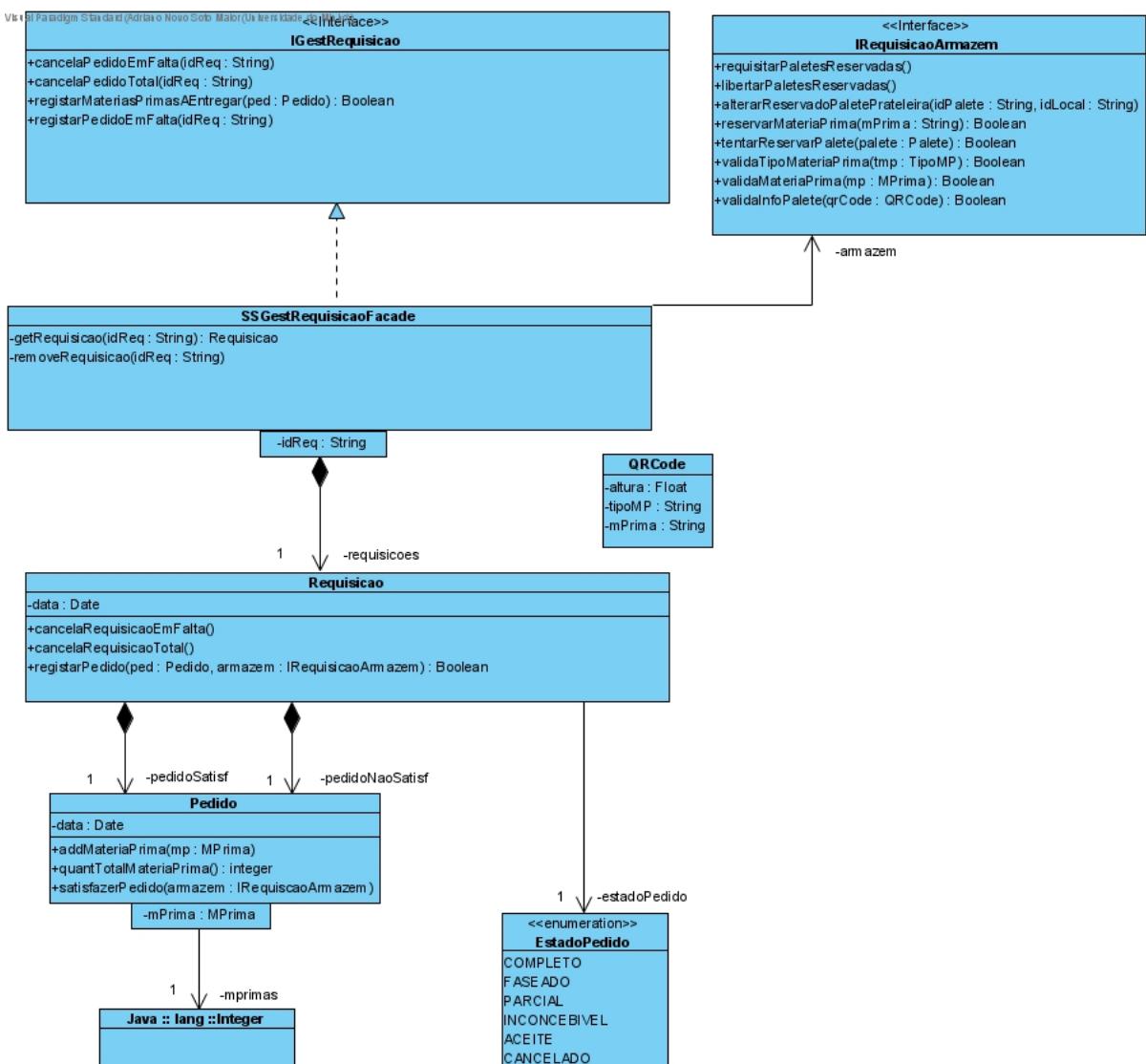
10.1. SSSArmazenamento



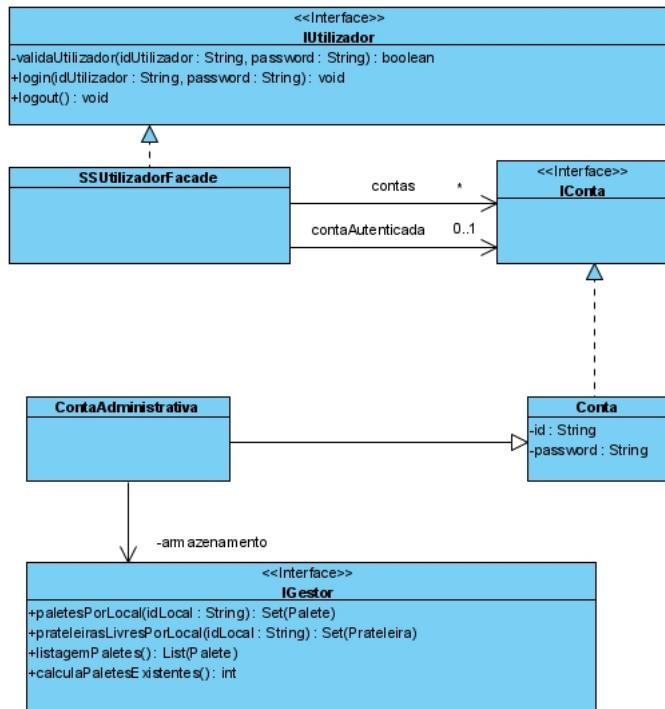
10.2. SSControloRobot



10.3. SRequisicao

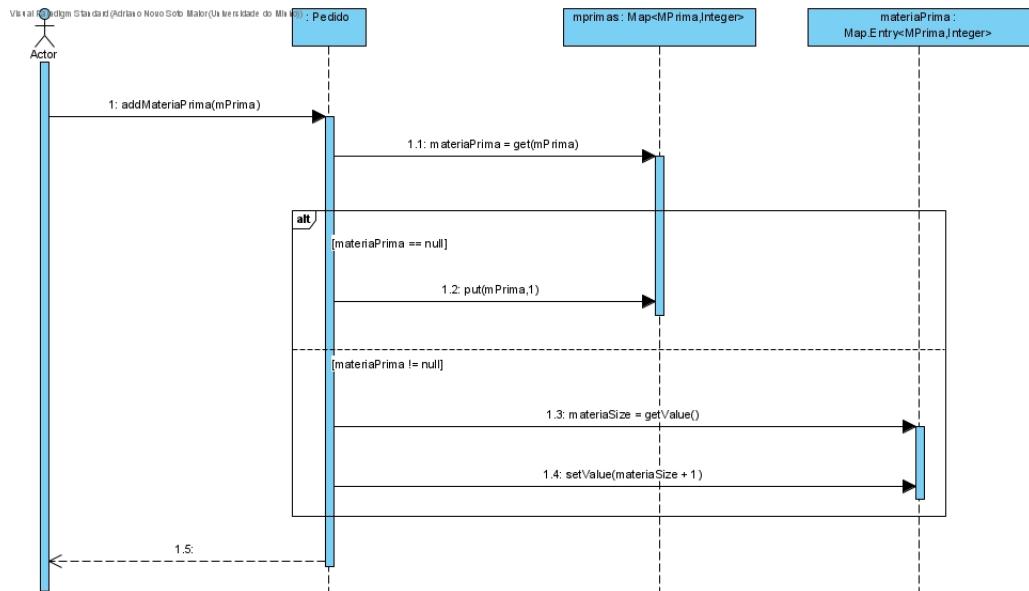


10.4. SSUtilizador

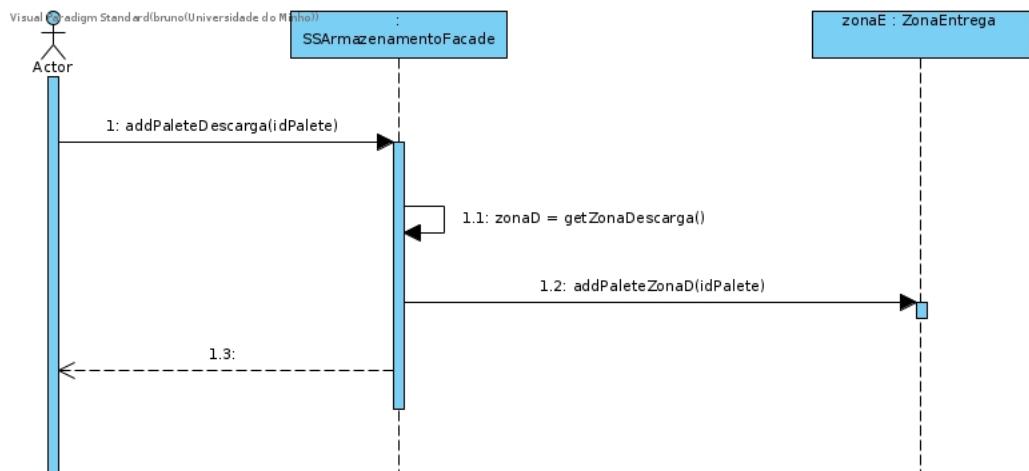


11 | Diagramas de Sequência

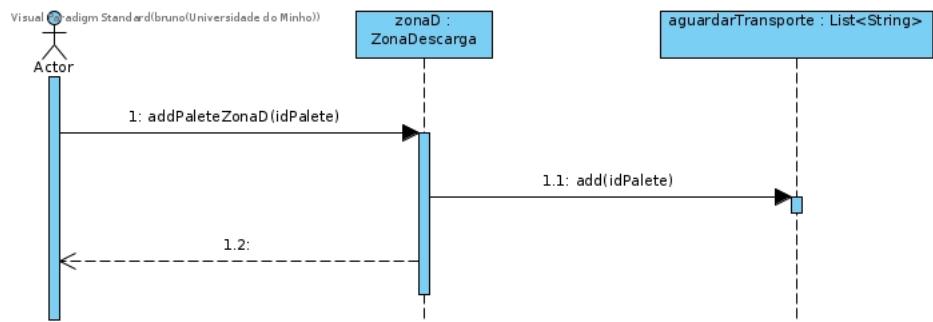
11.1. addMateriaPrima



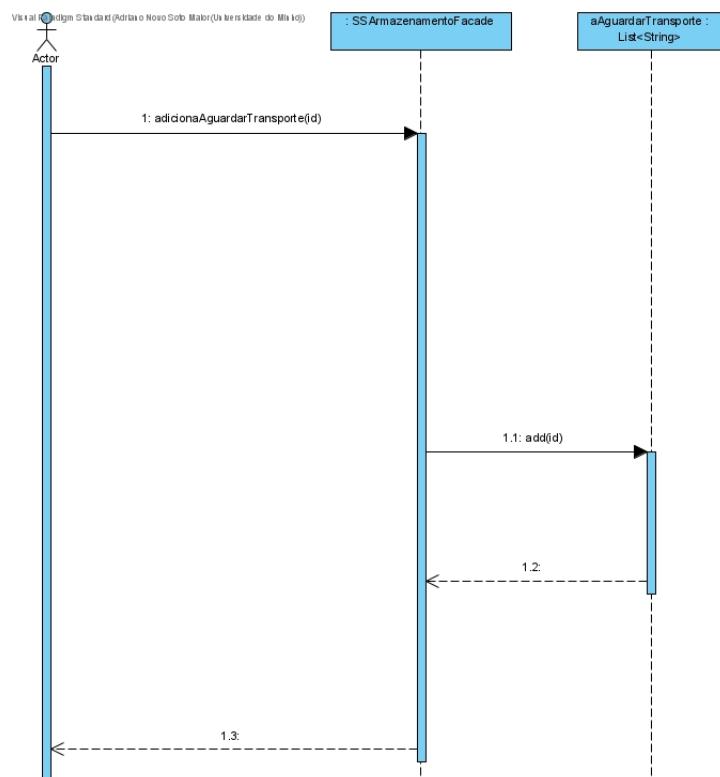
11.2. addPaleteDescarga



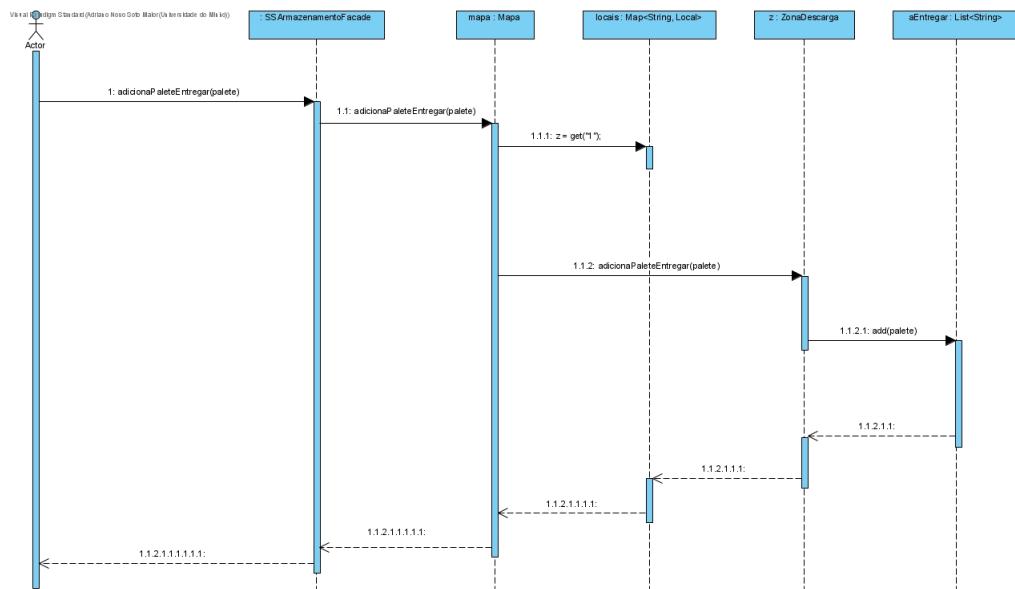
11.3. addPaleteZonaD



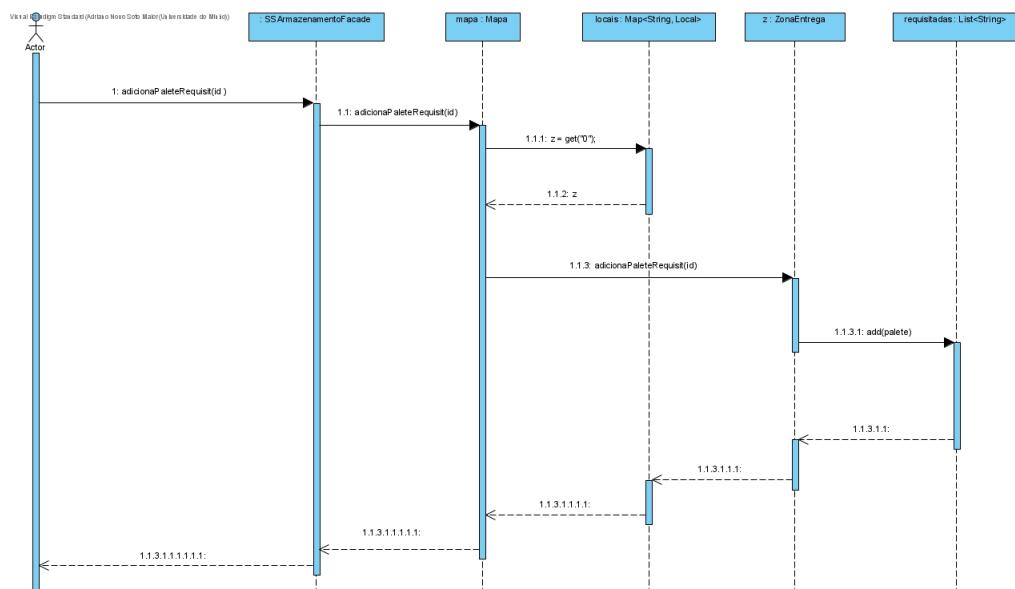
11.4. adicionaAguardarTransporte



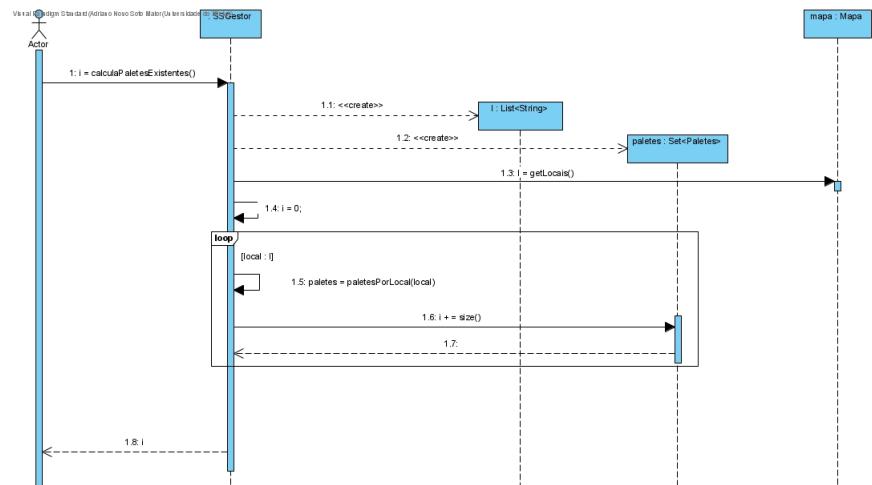
11.5. adicionaPaleteEntregar



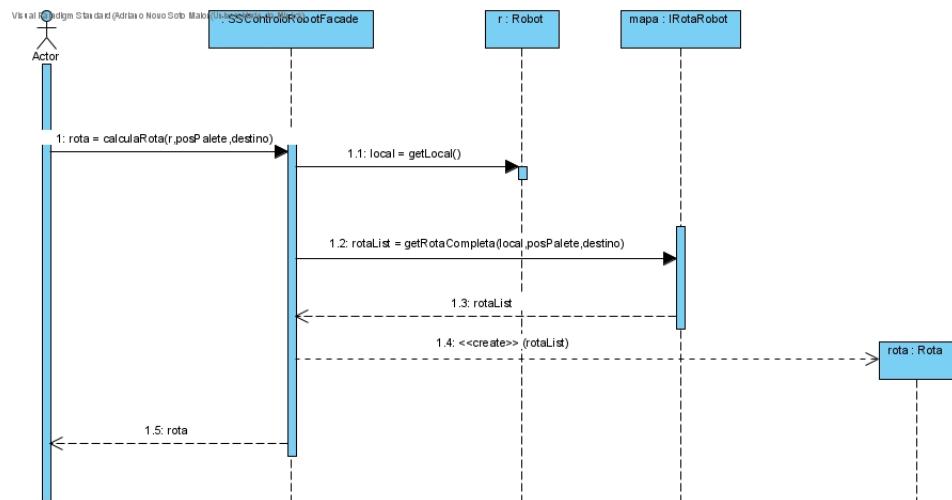
11.6. adicionaPaleteRequisit



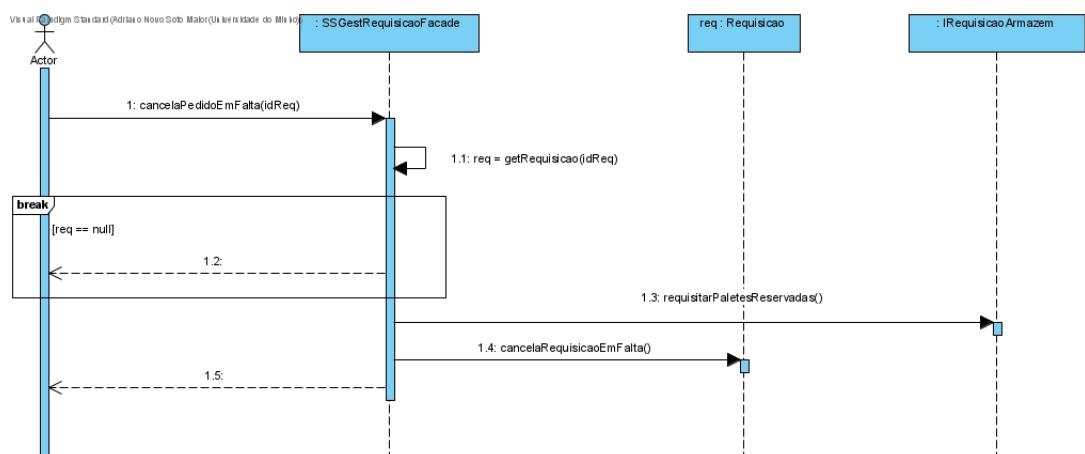
11.7. calculaPaletesExistentes



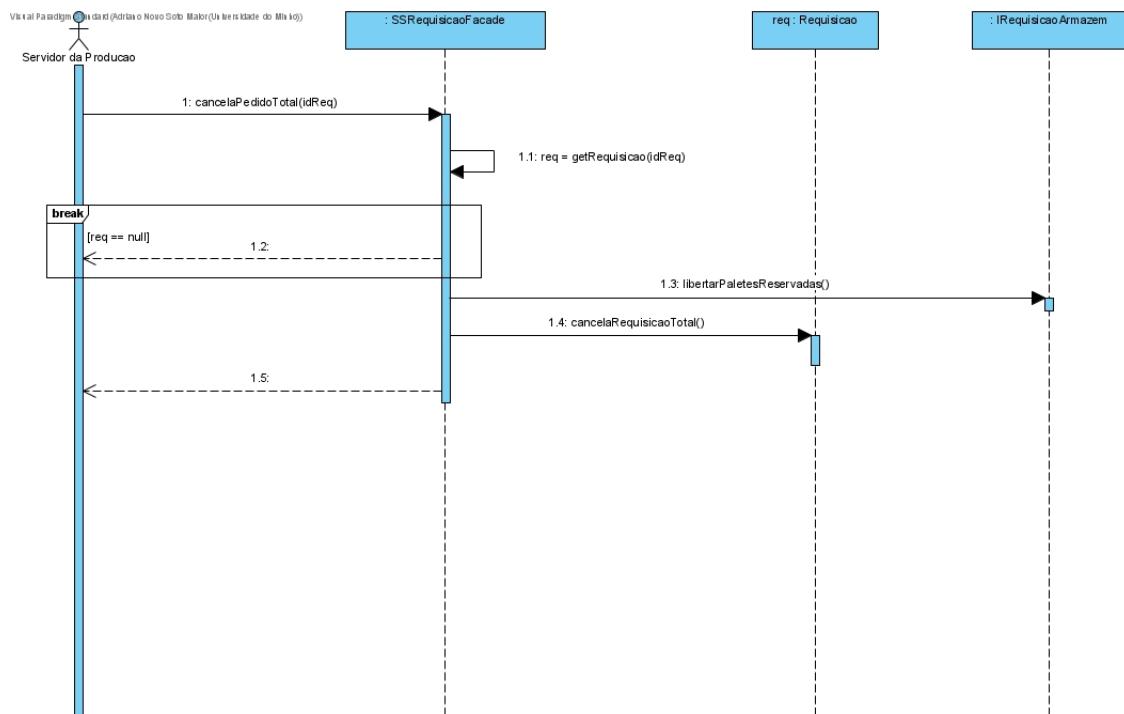
11.8. calculaRota



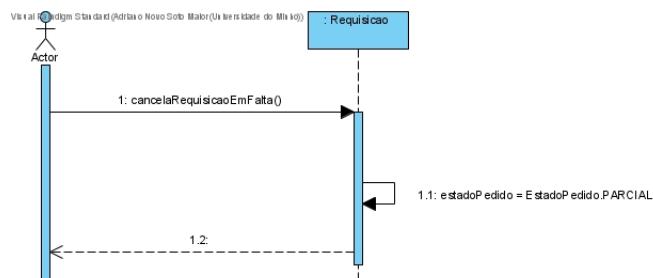
11.9. cancelaPedidoEmFalta



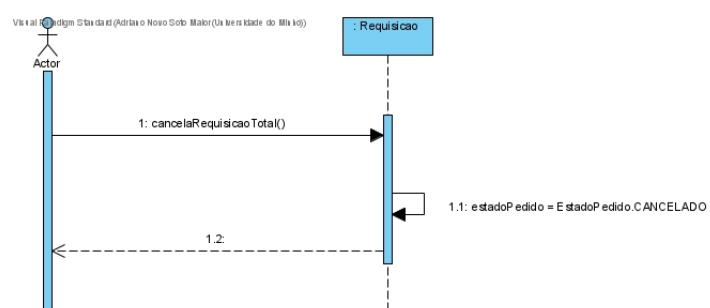
11.10. cancelaPedidoTotal



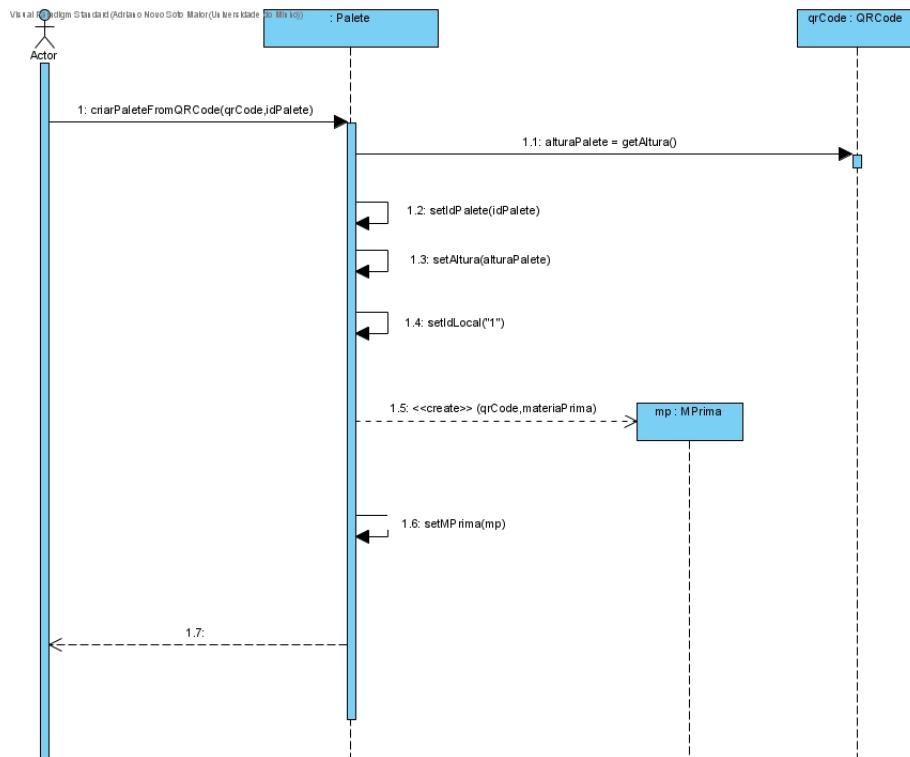
11.11. cancelaRequisicaoEmFalta



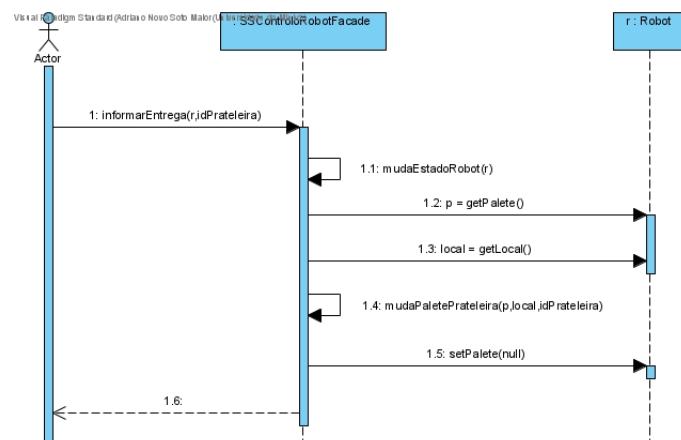
11.12. cancelaRequisicaoTotal



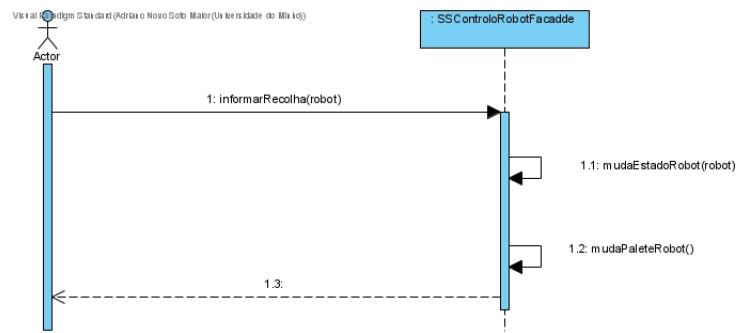
11.13. criarPaleteFromQRCode



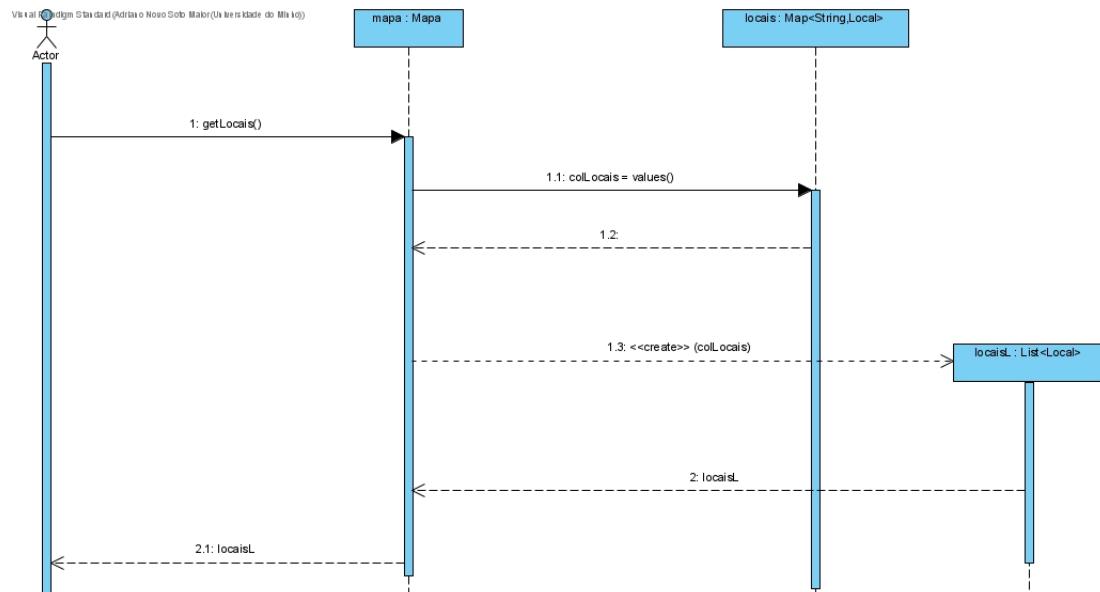
11.14. informarEntrega



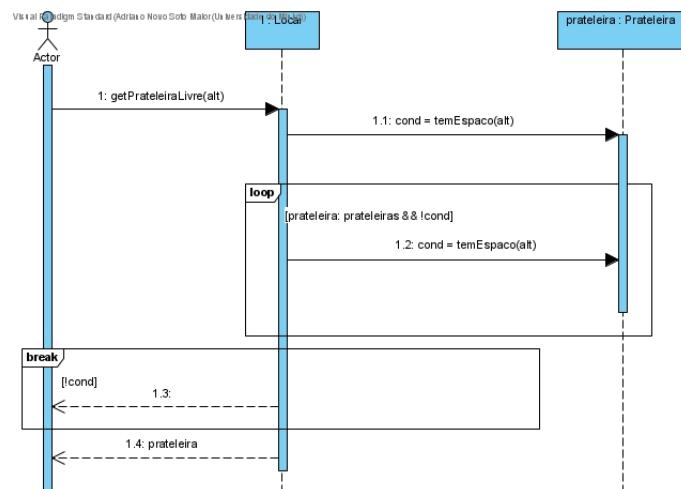
11.15. informarRecolha



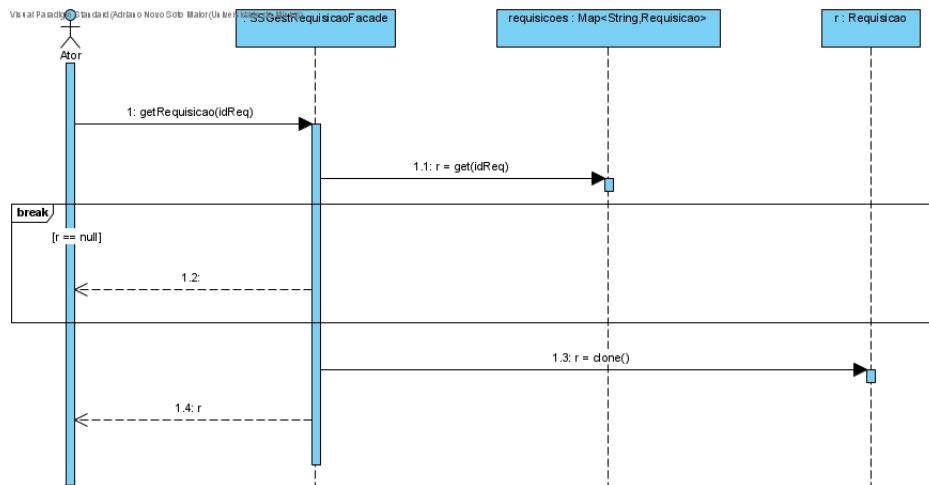
11.16. getLocais



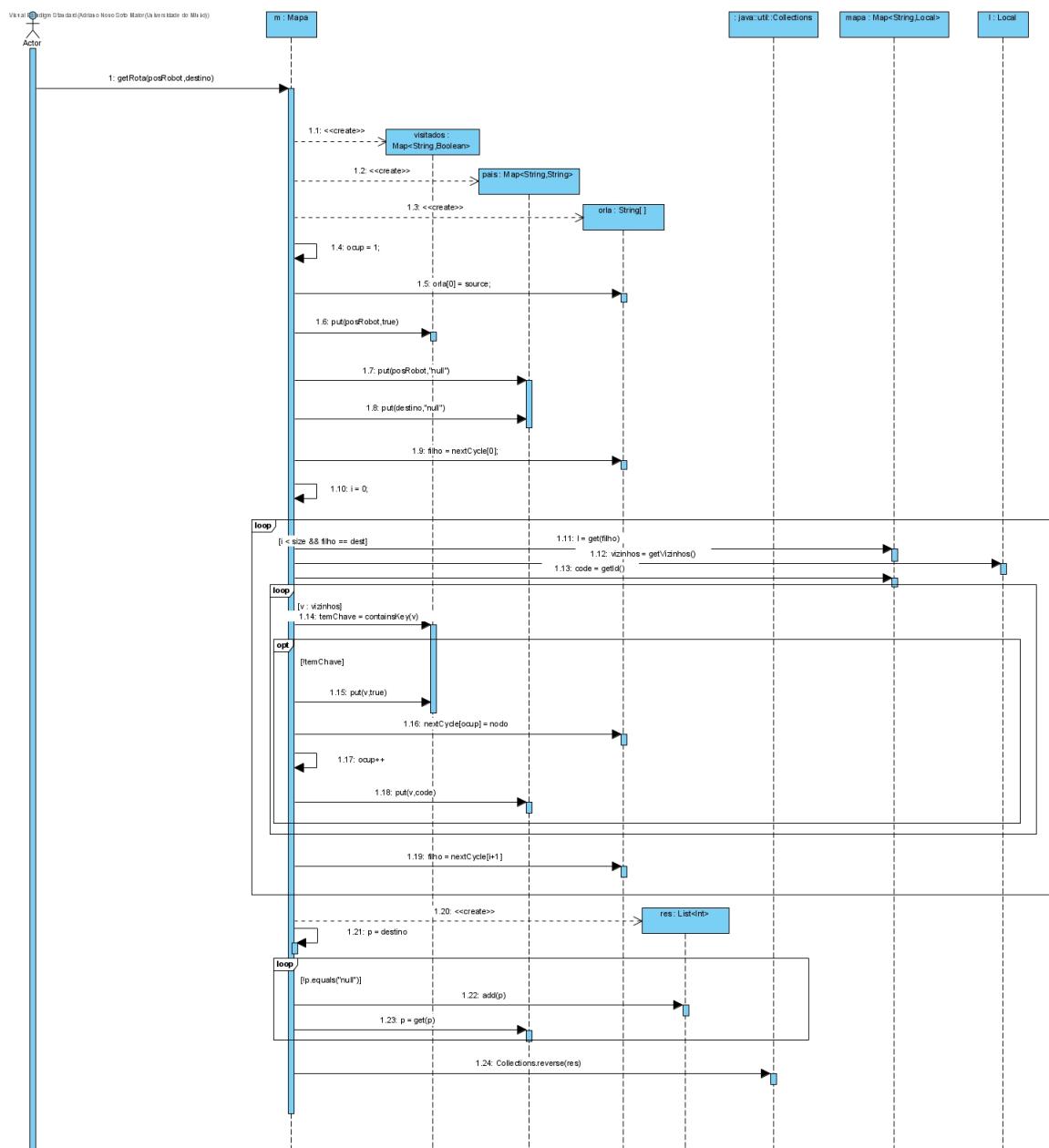
11.17. getPrateleiraLivre



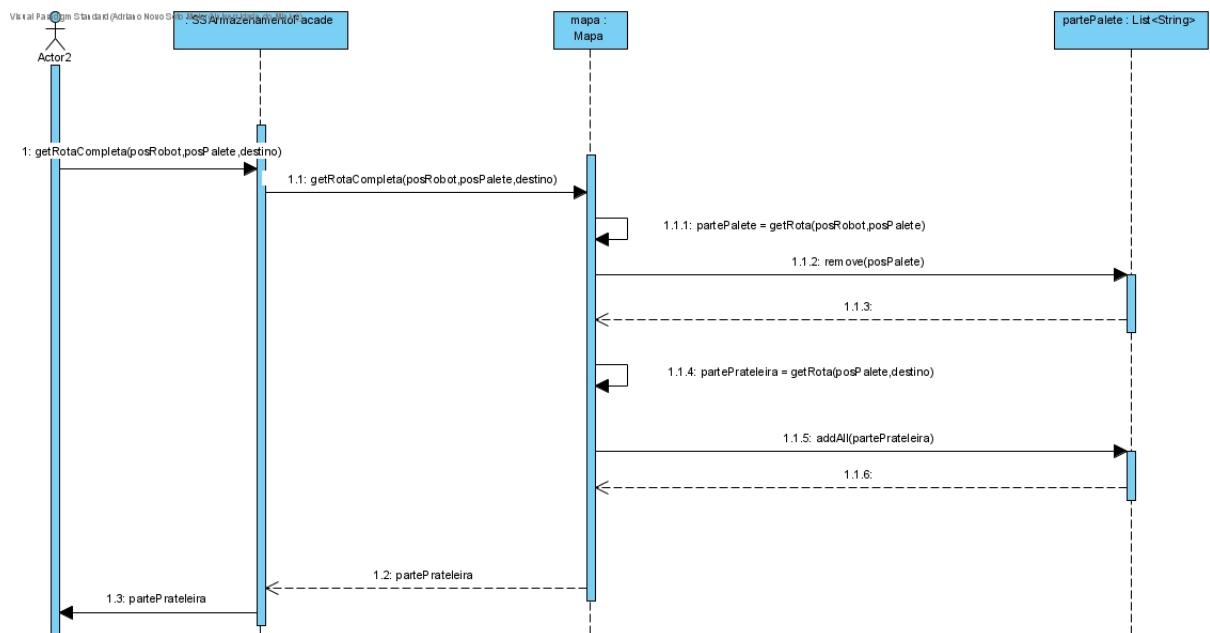
11.18. getRequisicao



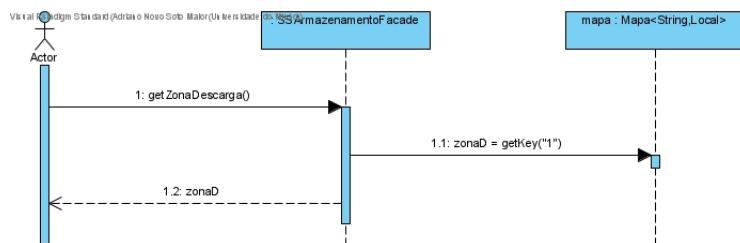
11.19. getRota



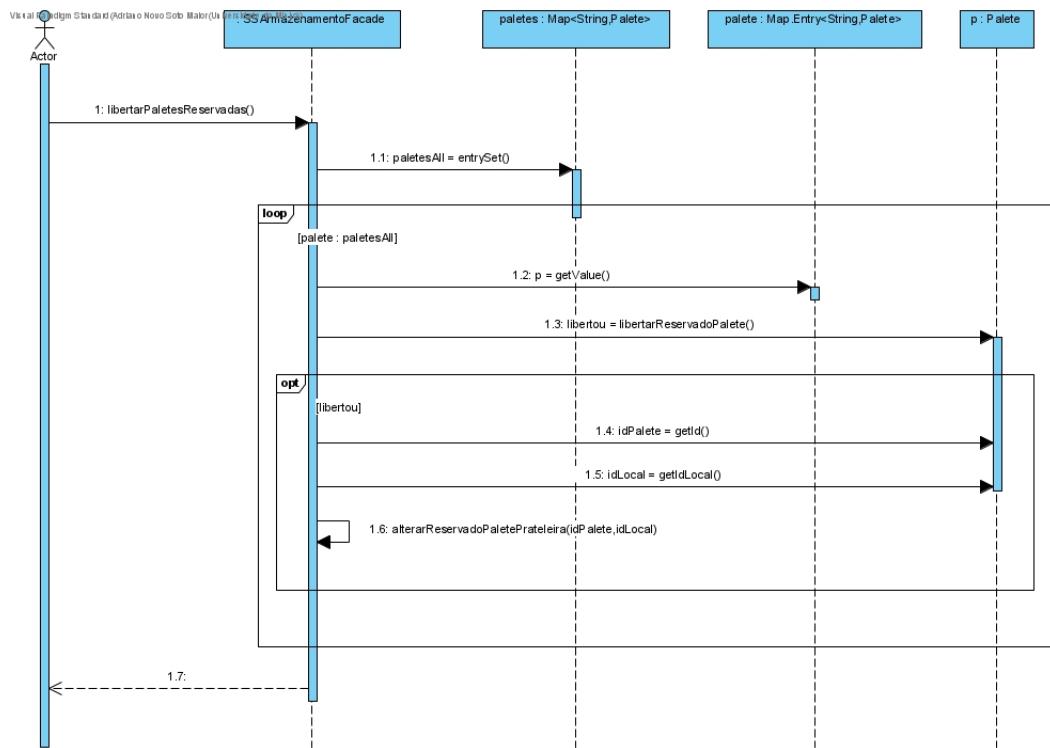
11.20. getRotaCompleta



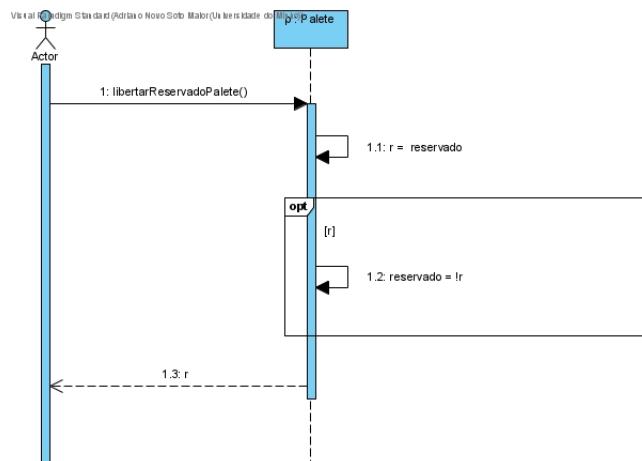
11.21. getZonaDescarga



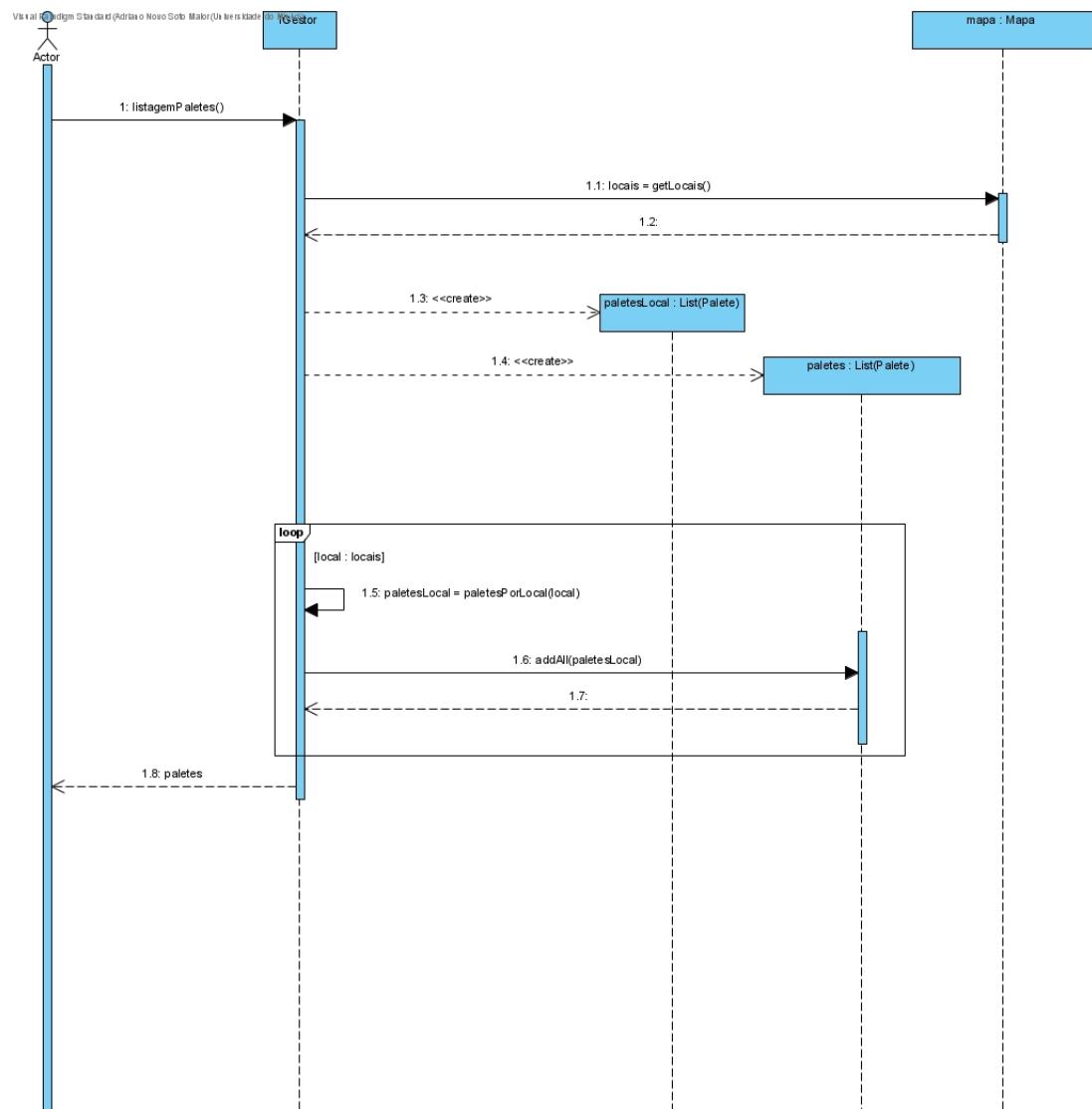
11.22. libertarPaletesReservadas



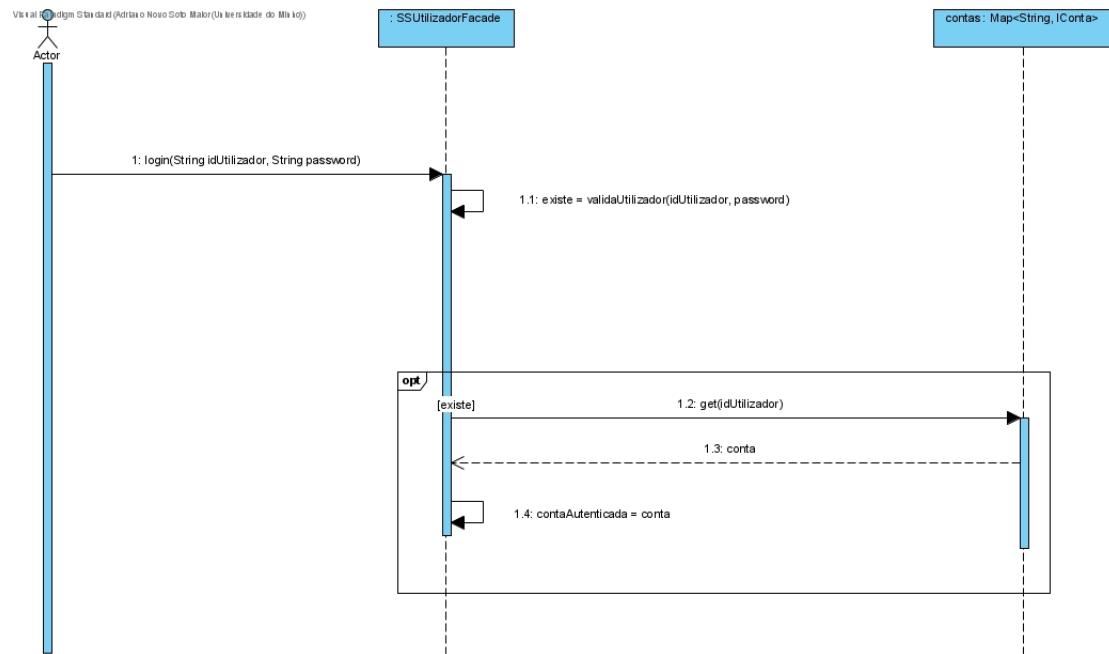
11.23. libertarReservadoPaleta



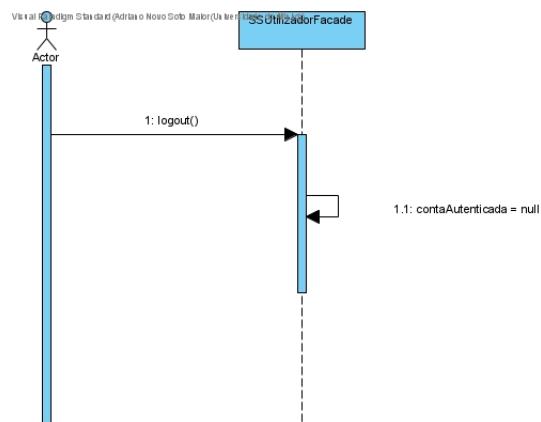
11.24. listagemPaletes



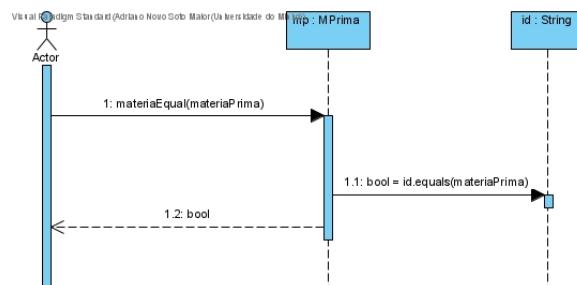
11.25. login



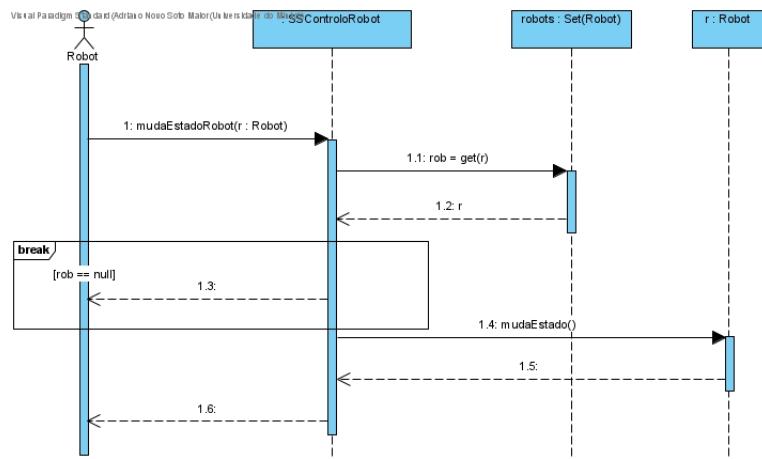
11.26. logout



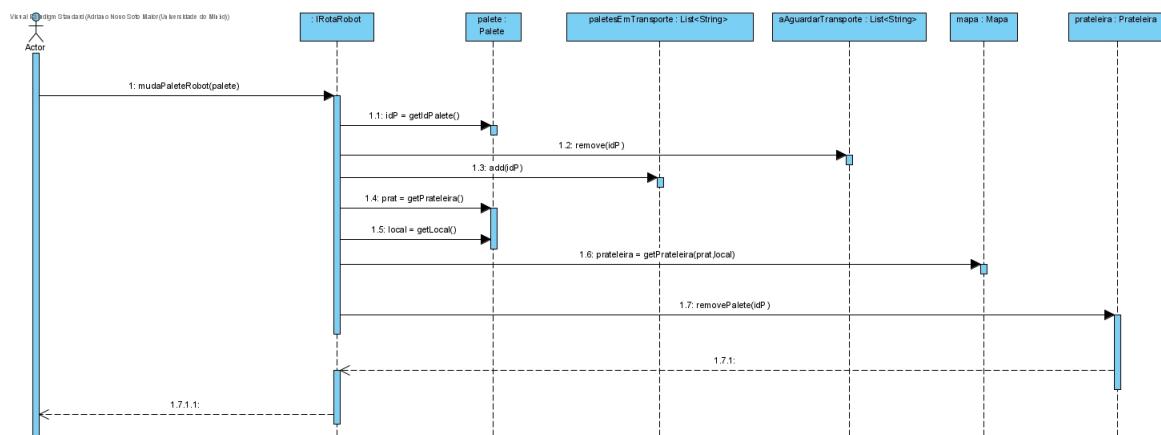
11.27. materiaEqual



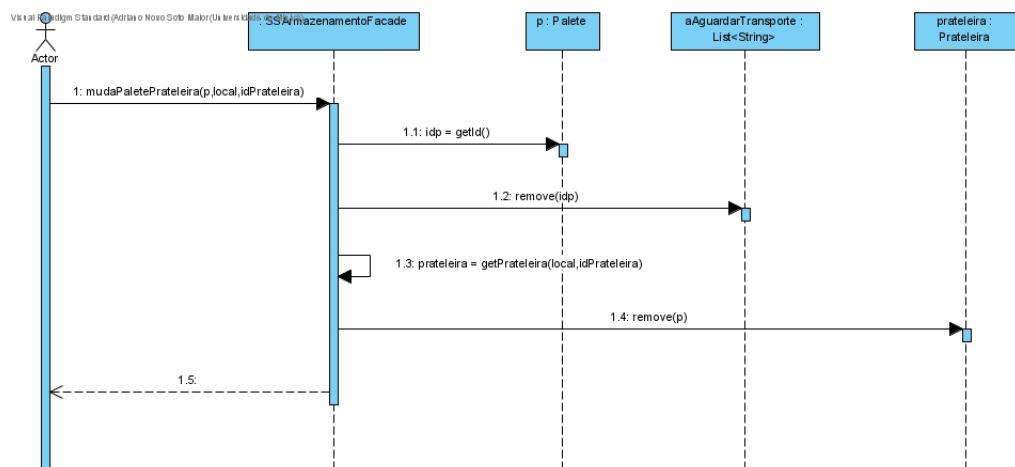
11.28. mudaEstadoRobot



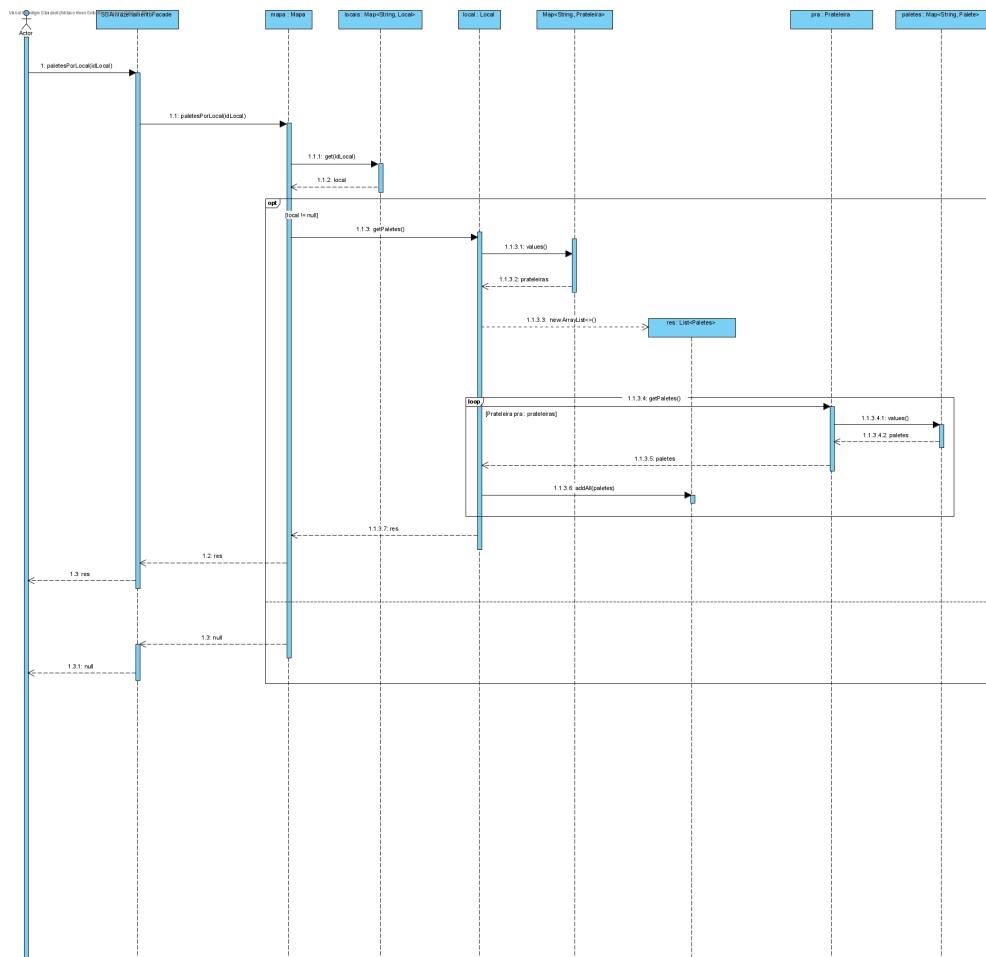
11.29. mudaPaleteRobot



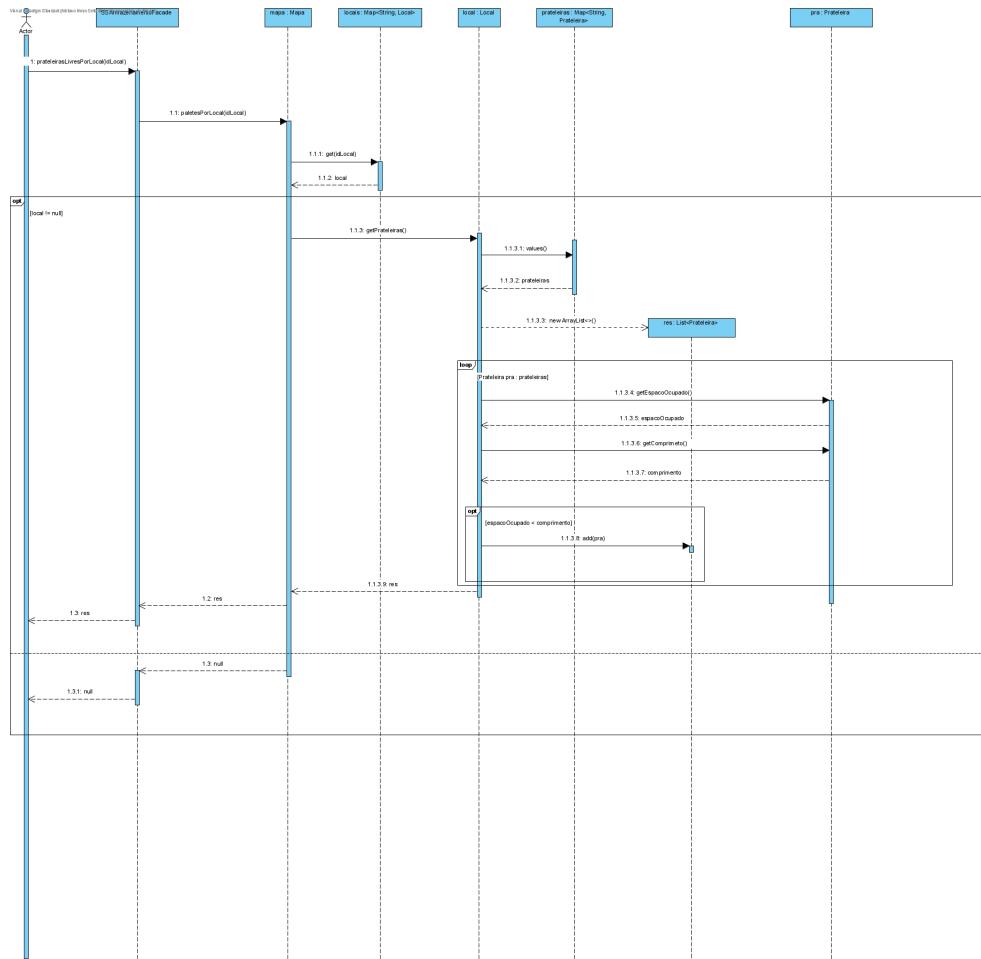
11.30. mudaPaletePrateleira



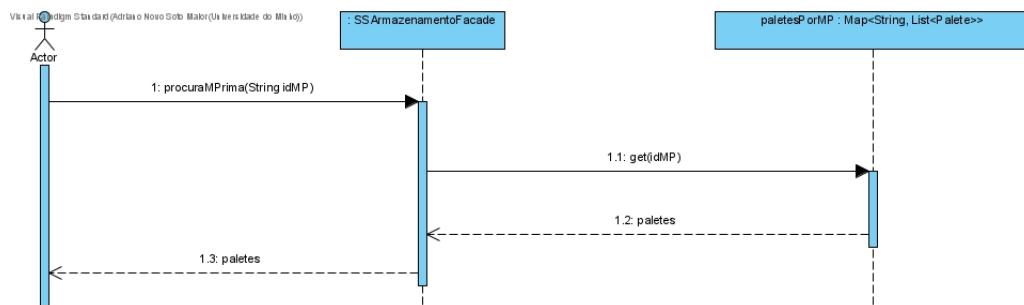
11.31. paletesPorLocal



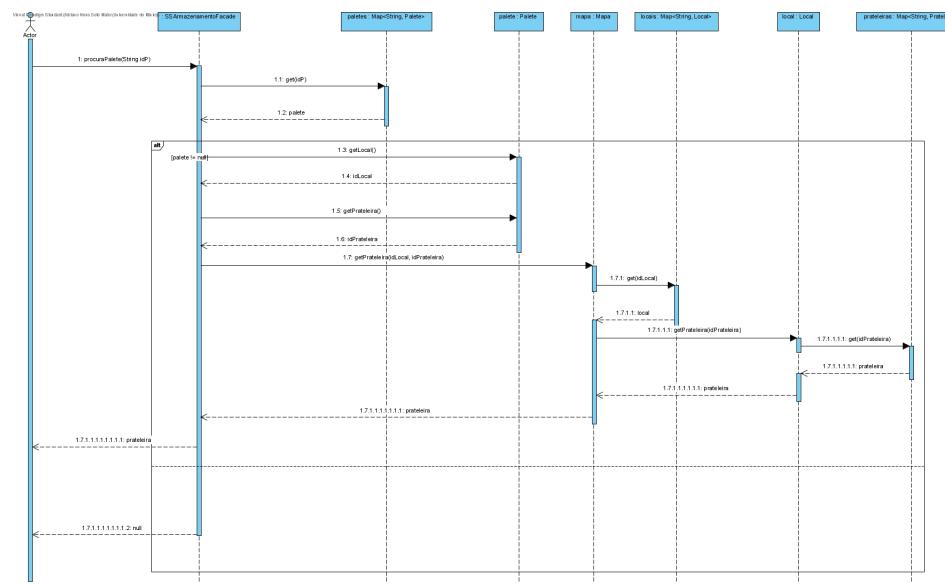
11.32. prateleirasLivresPorLocal



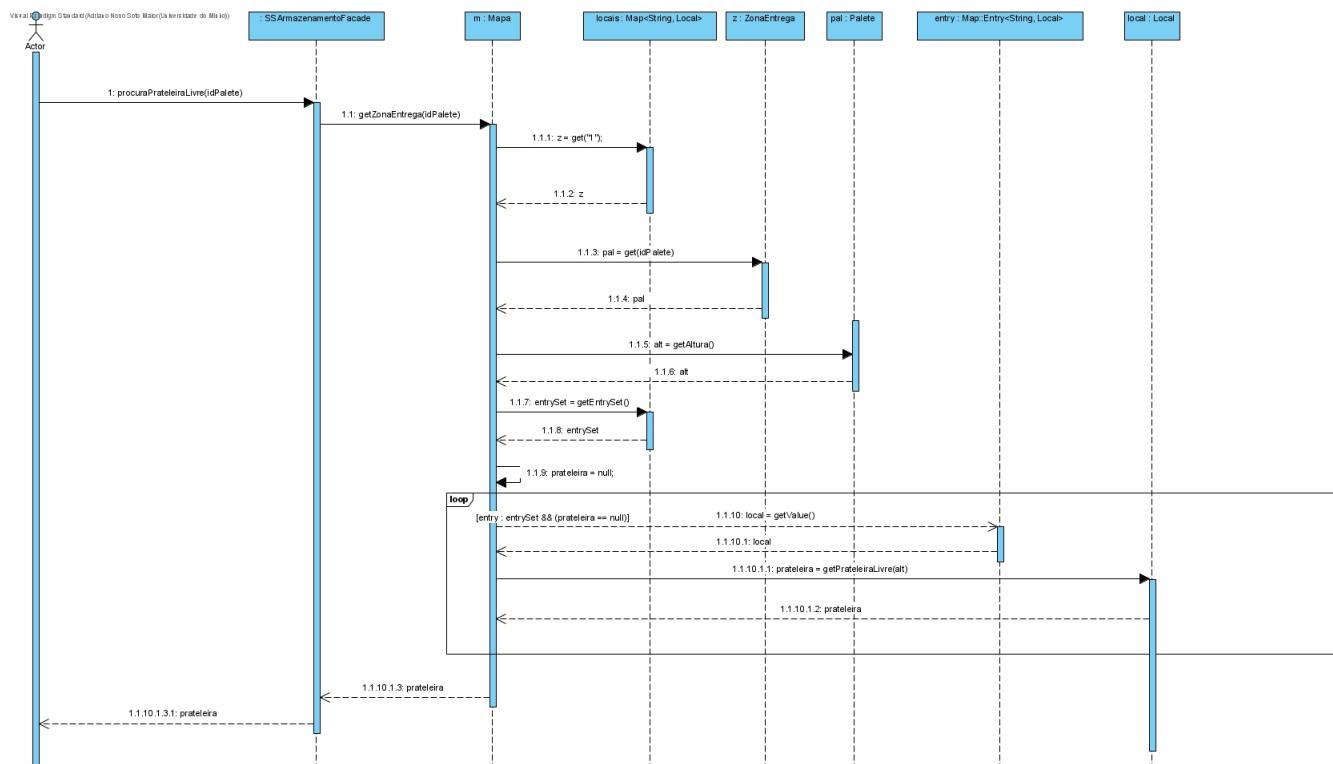
11.33. procuraMPrima



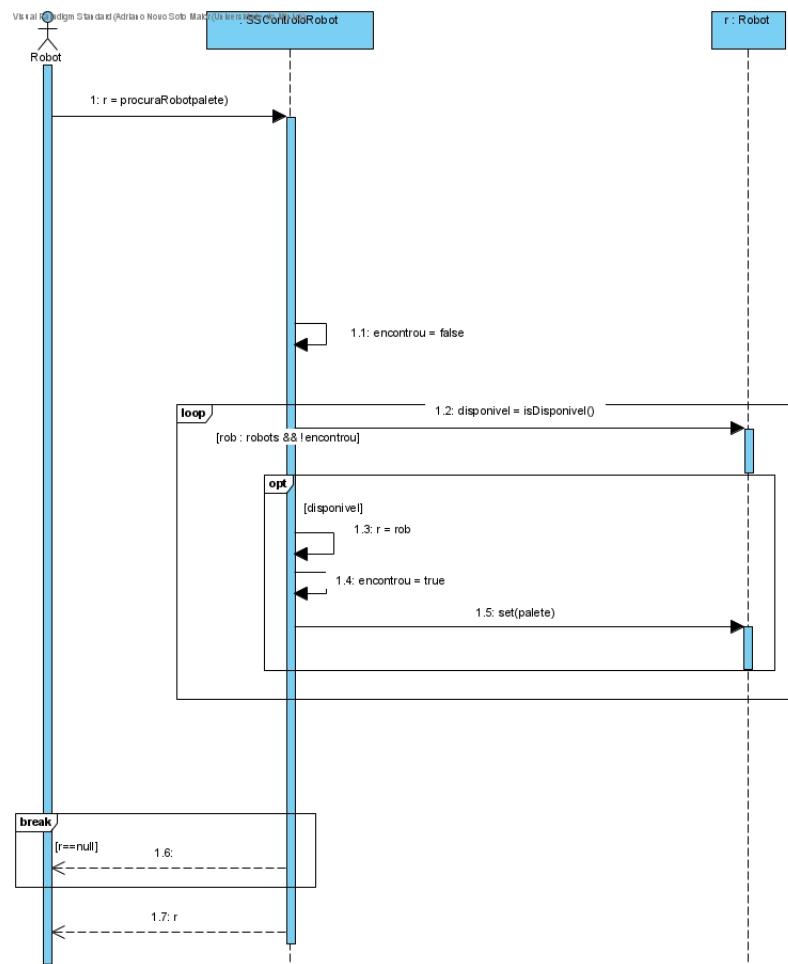
11.34. procuraPalete



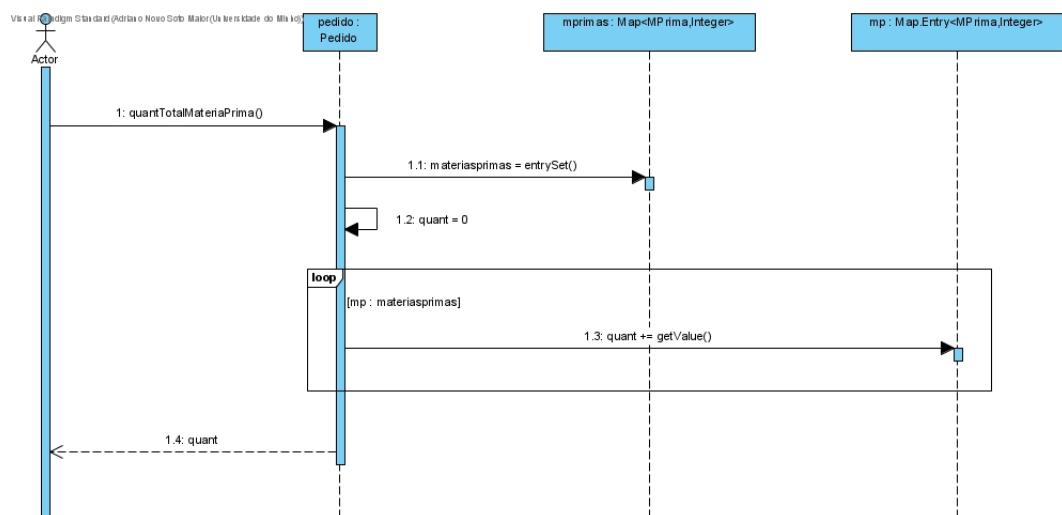
11.35. procuraPrateleiraLivre



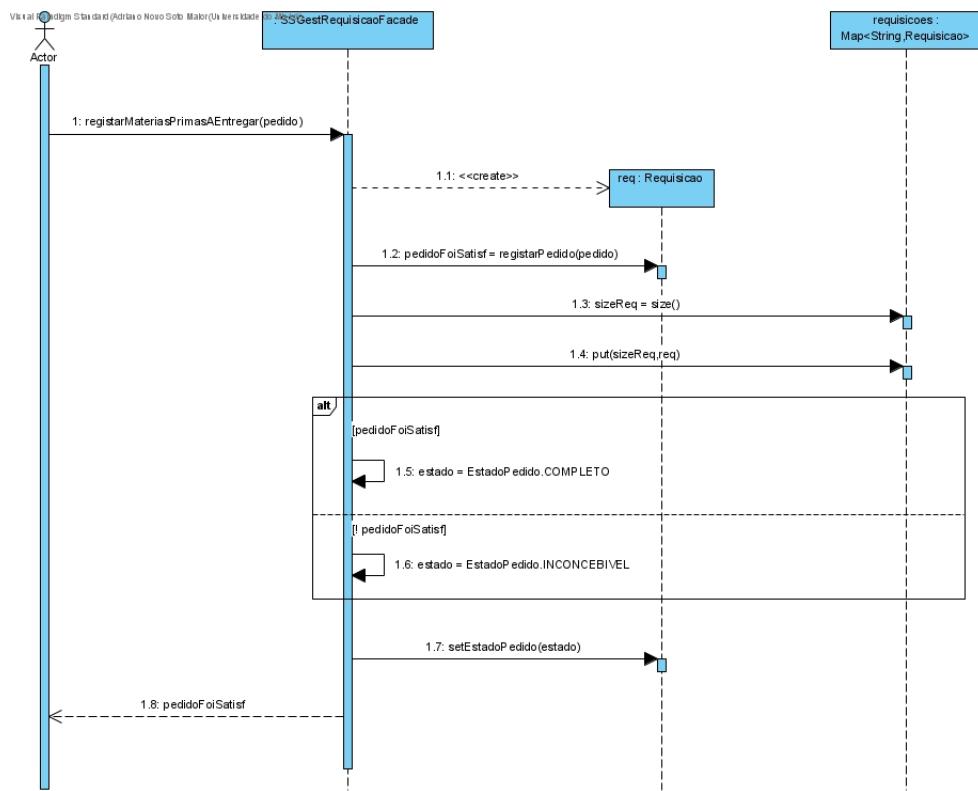
11.36. procuraRobot



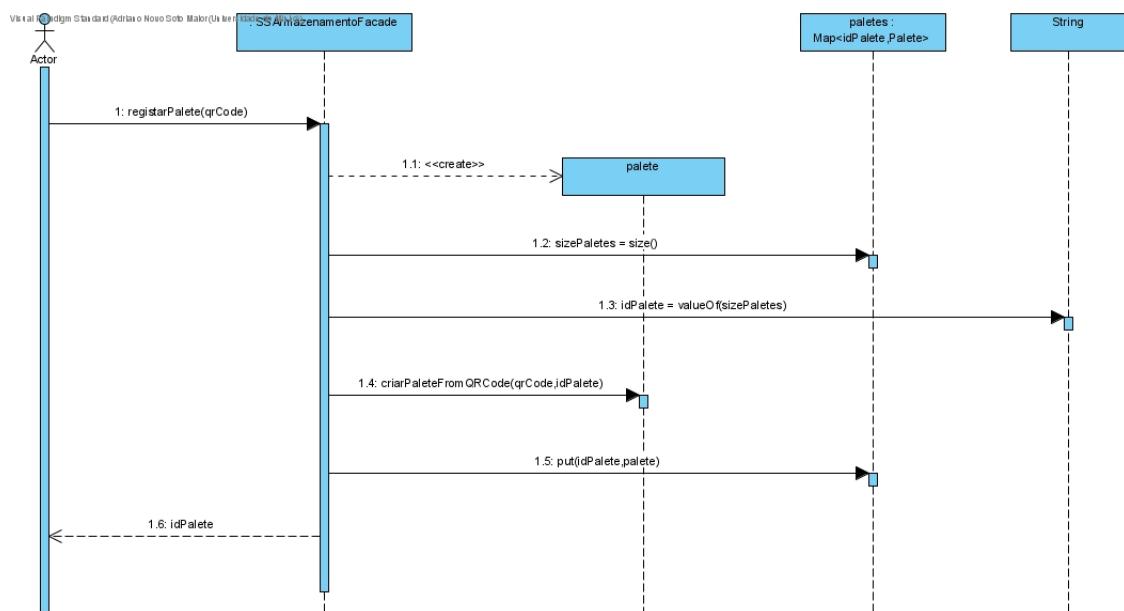
11.37. quantTotalMateriaPrima



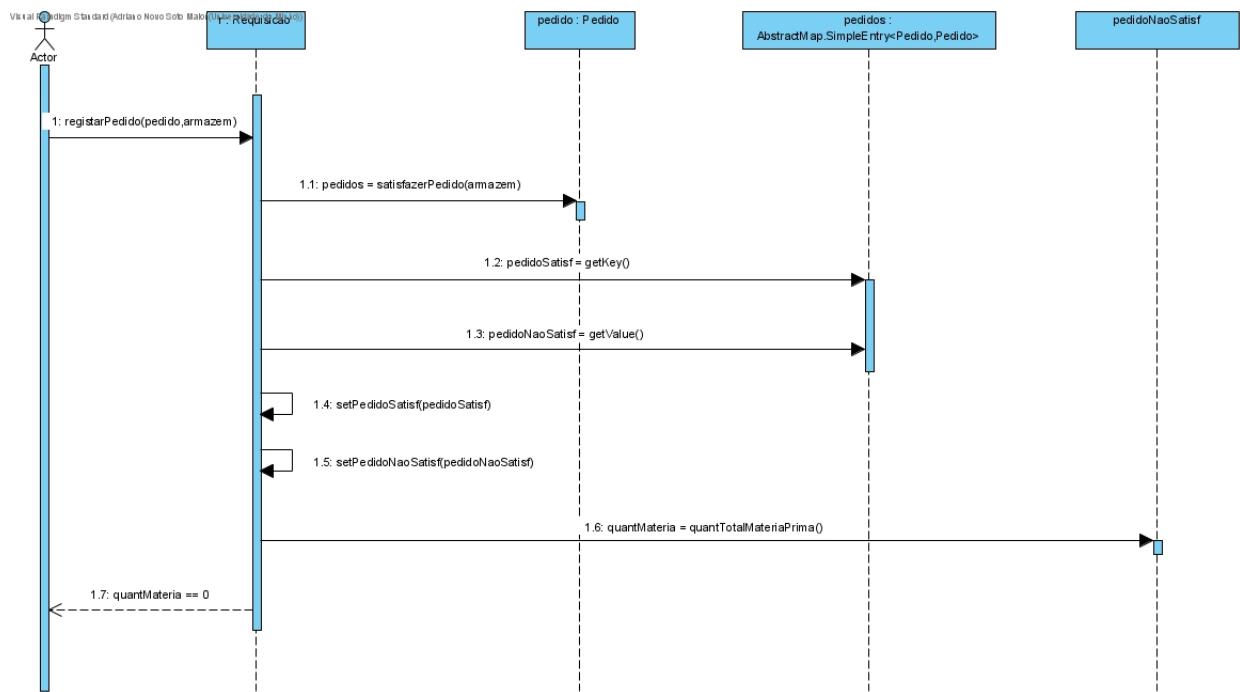
11.38. registrarMateriasPrimasAEntregar



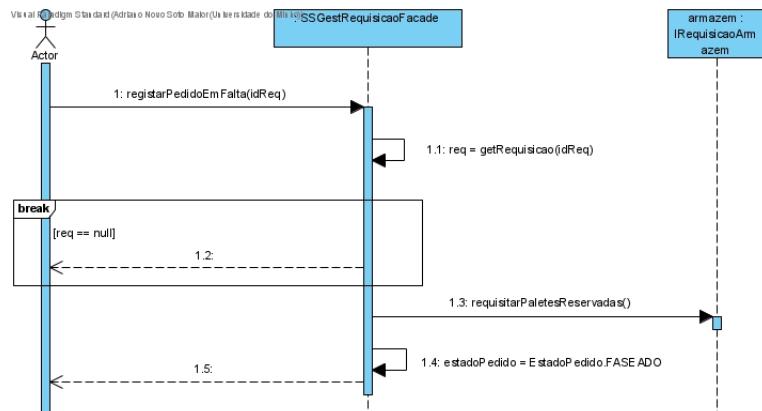
11.39. registrarPalete



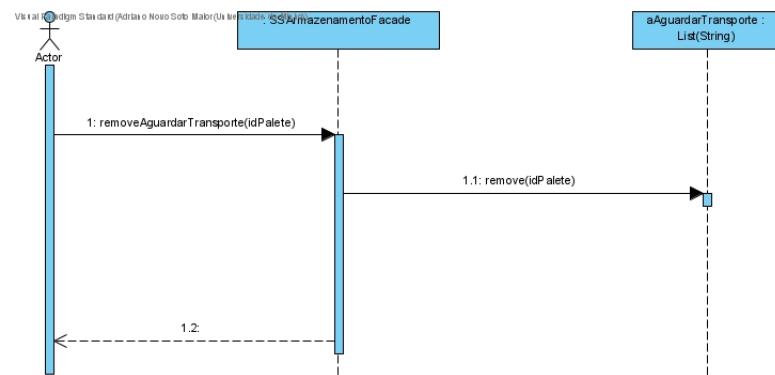
11.40. registrarPedido



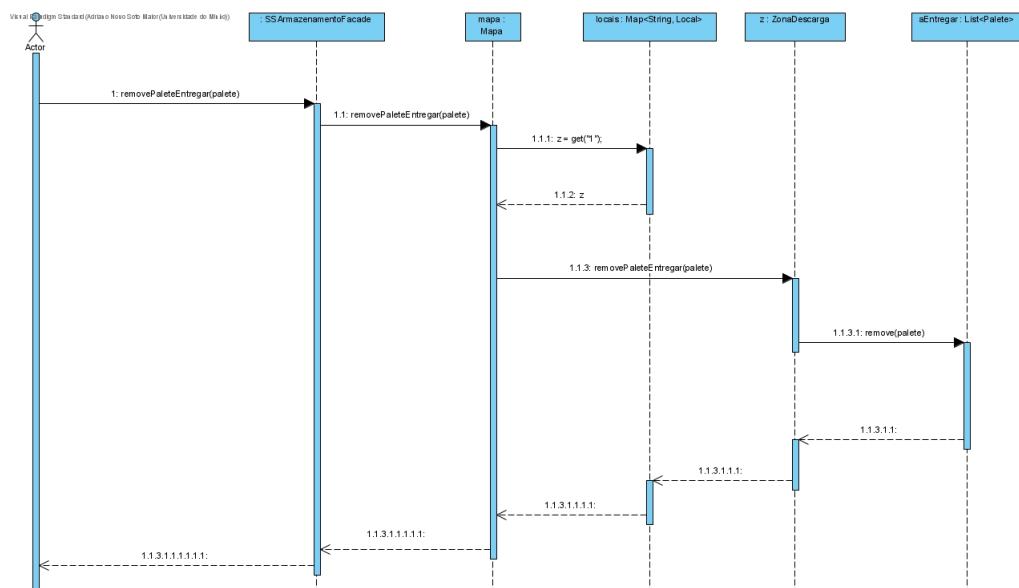
11.41. registrarPedidoEmFalta



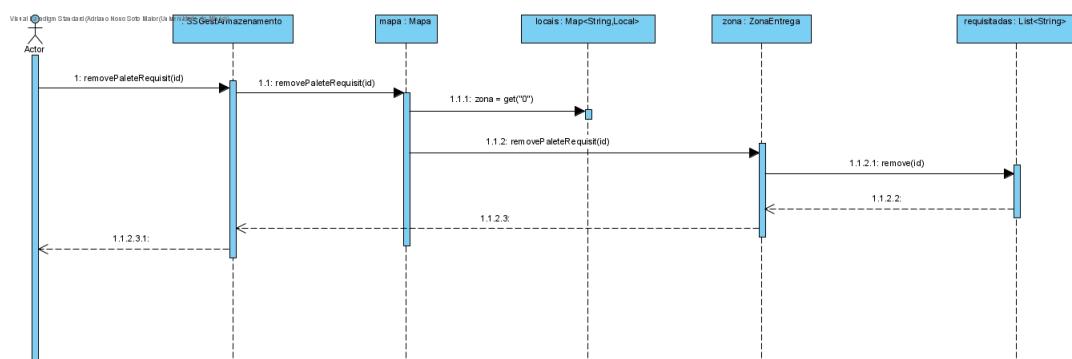
11.42. removeAguardarTransporte



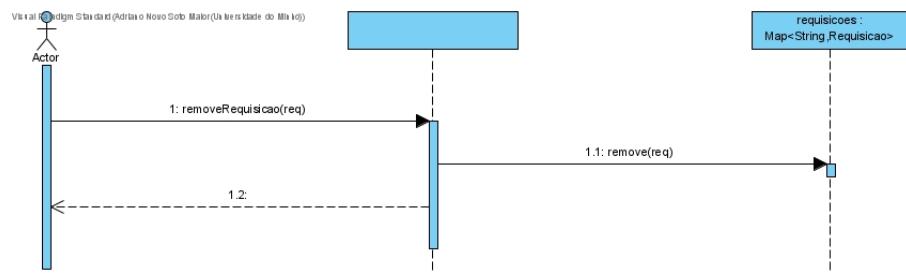
11.43. removePaleteEntregar



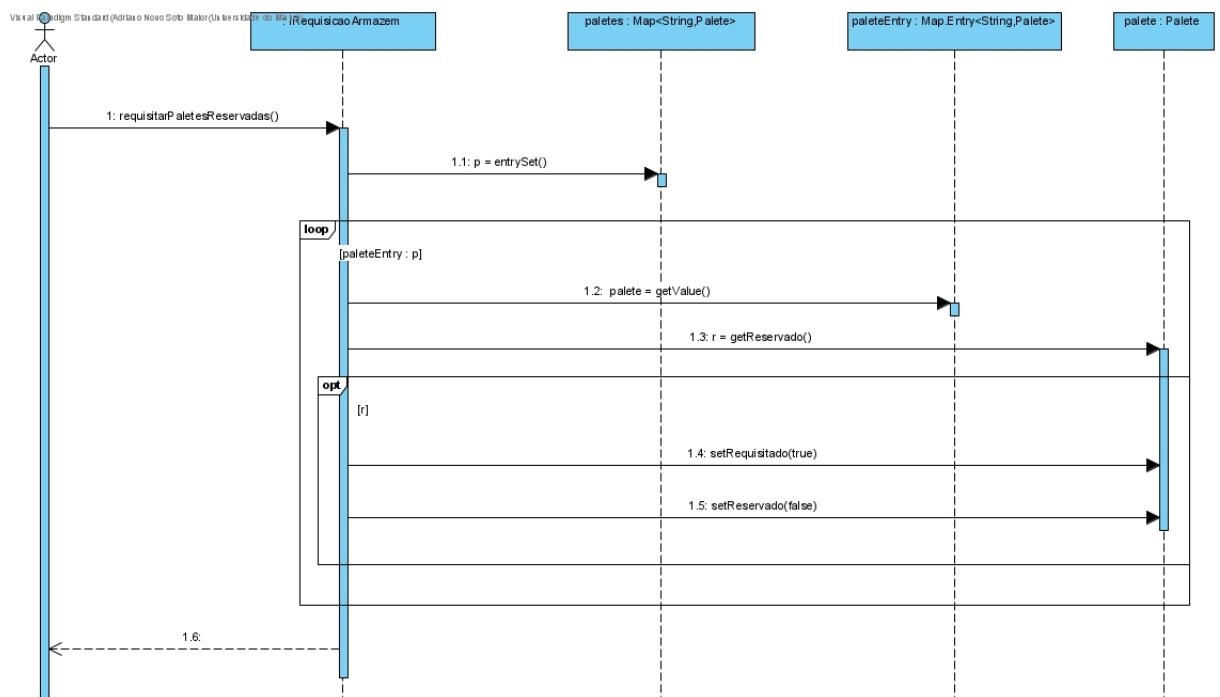
11.44. removePaleteRequisit



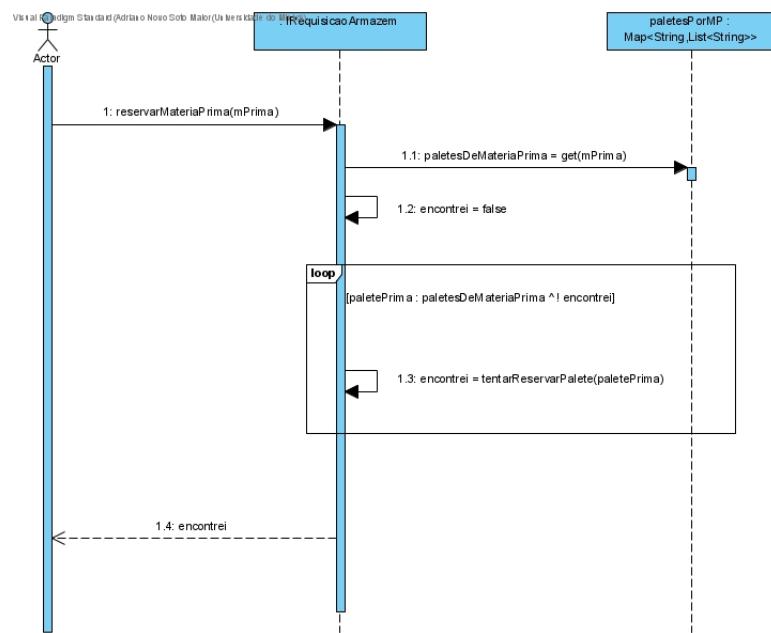
11.45. removeRequisicao



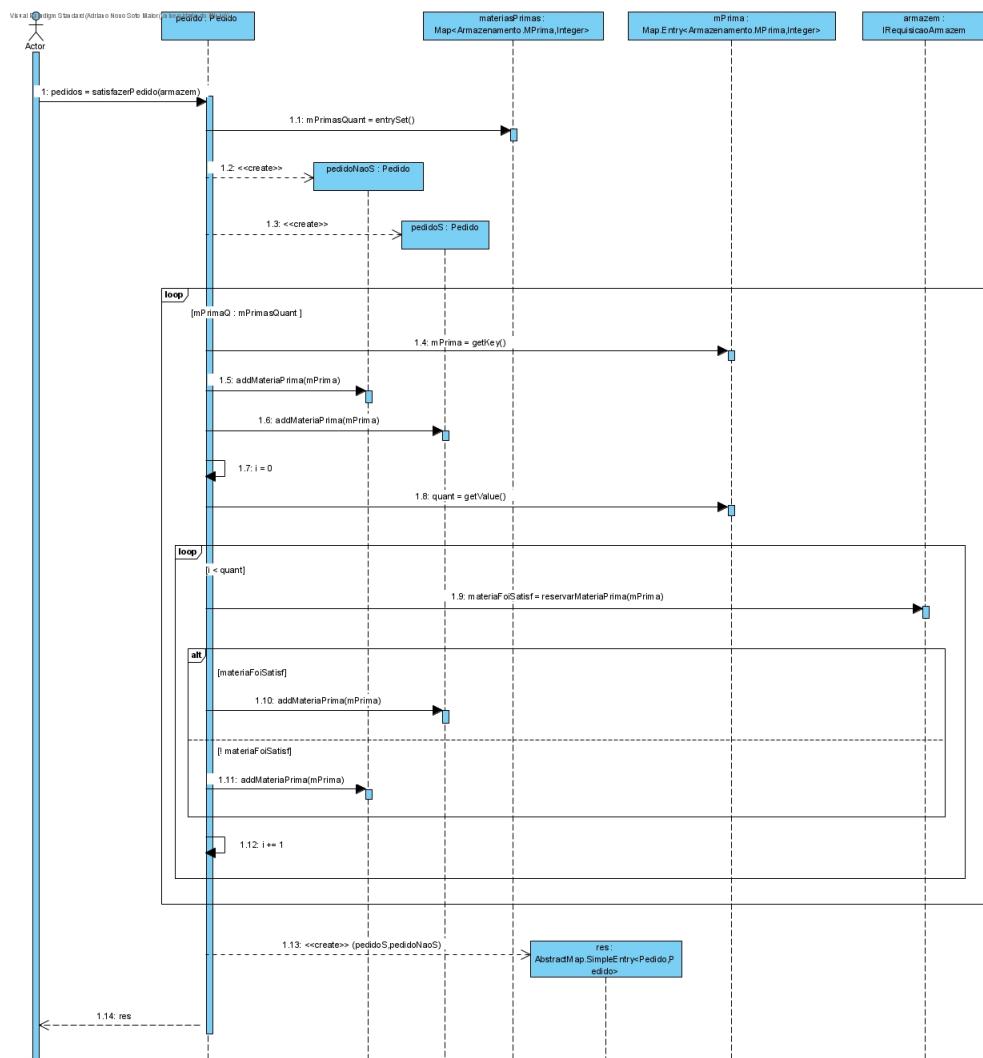
11.46. requisitarPaletesReservadas



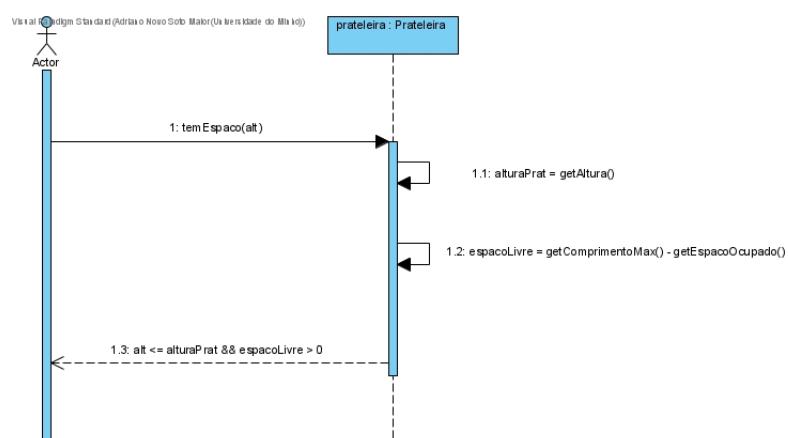
11.47. reservarMateriaPrima



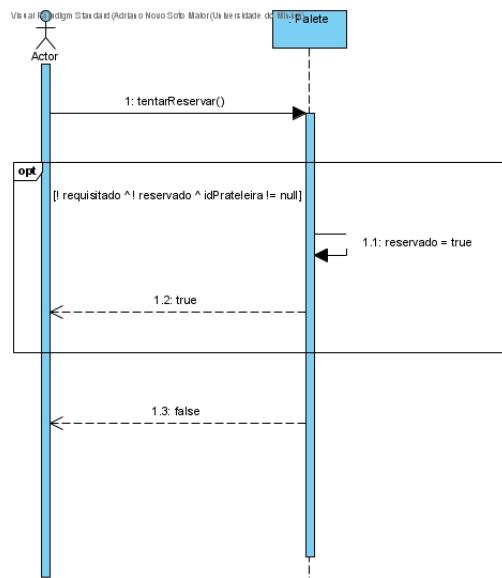
11.48. satisfazerPedido



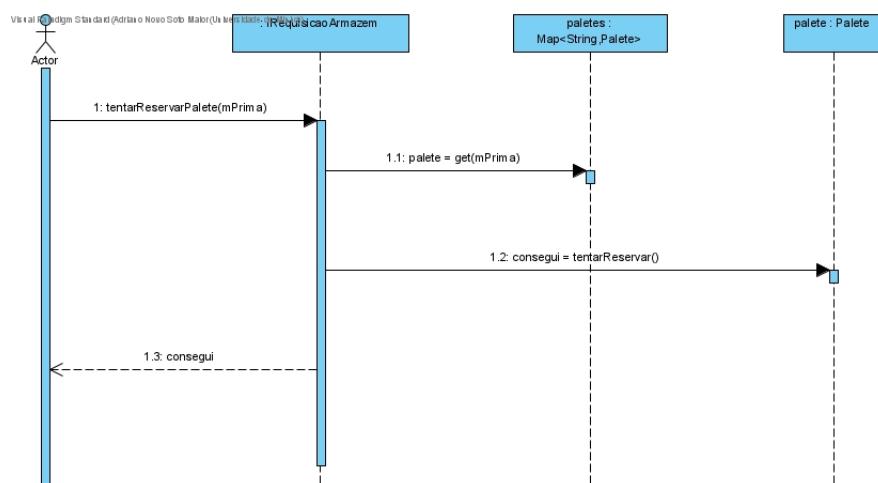
11.49. temEspaco



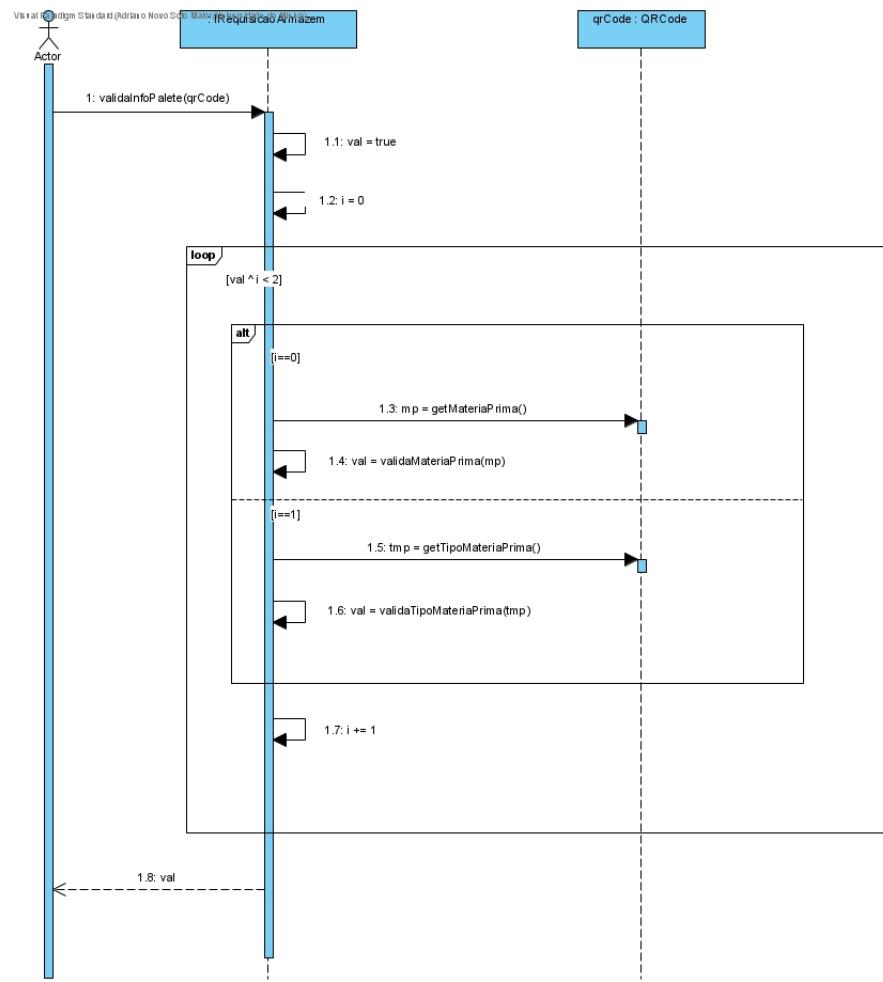
11.50. tentarReservar



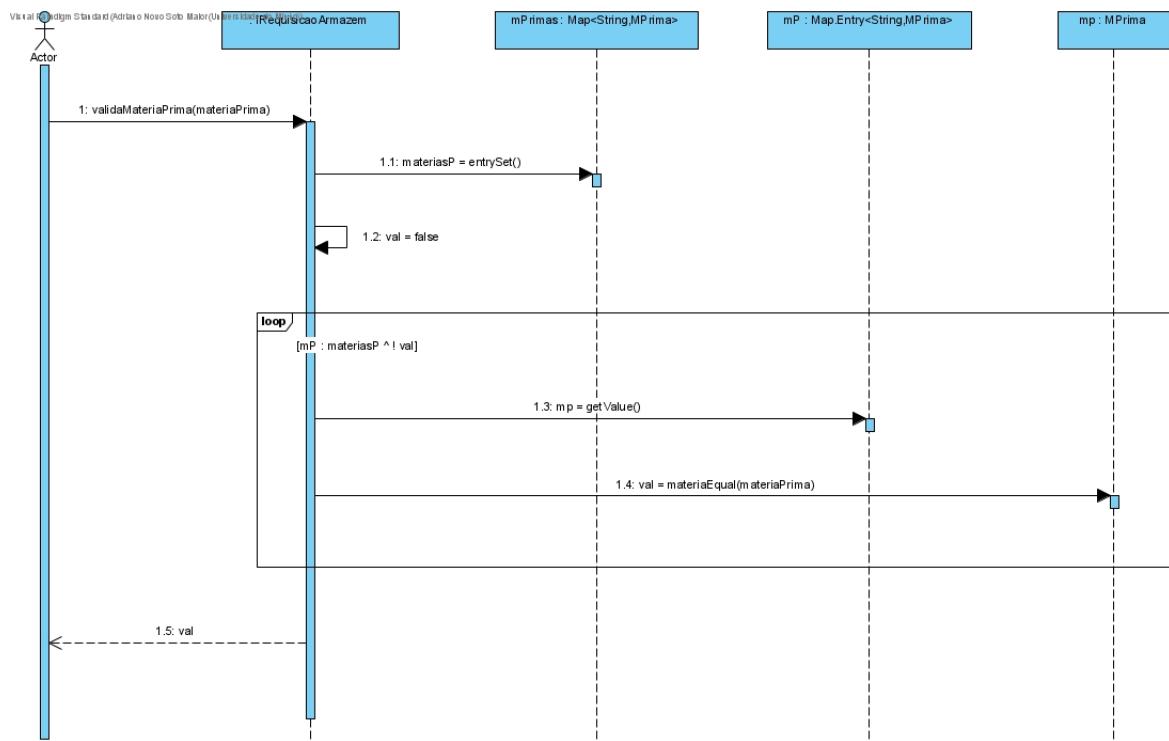
11.51. tentarReservarPalete



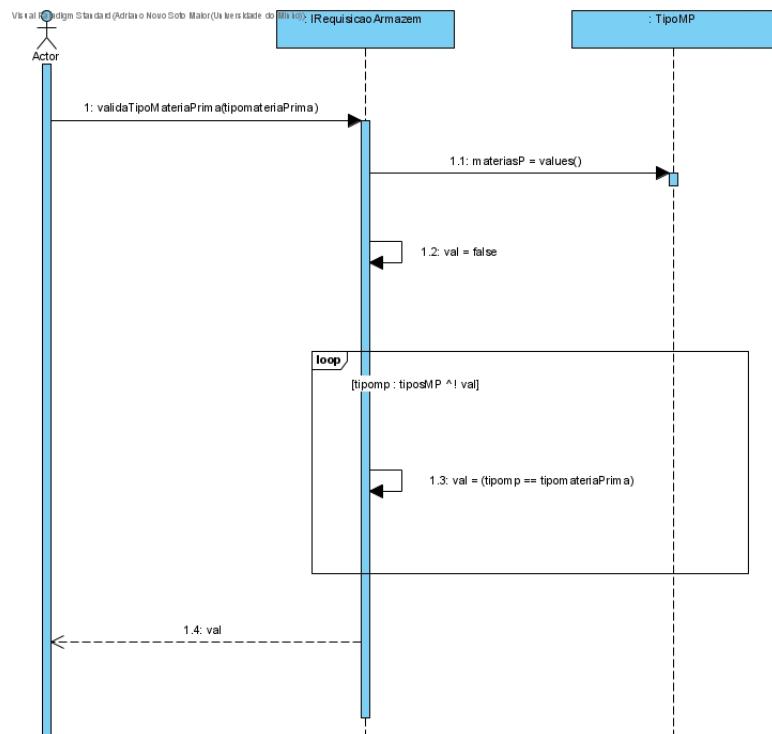
11.52. validaInfoPalete



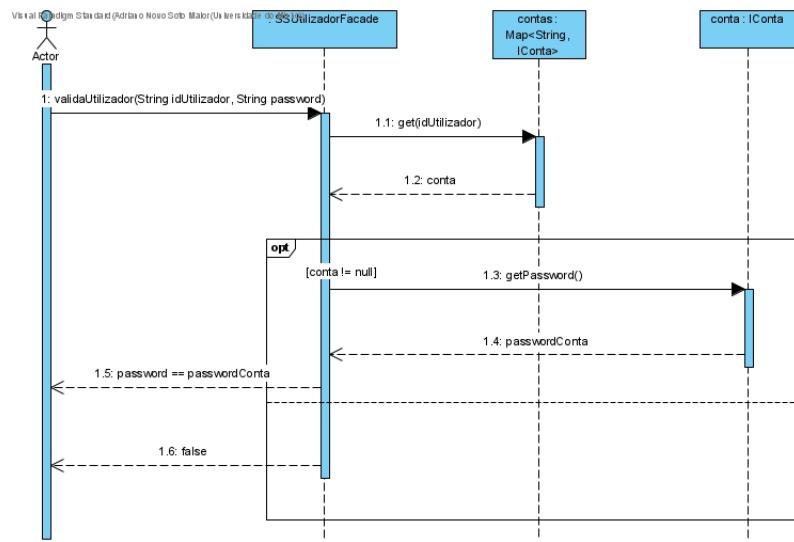
11.53. validaMateriaPrima



11.54. validaTipoMateriaPrima



11.55. validaUtilizador



12 | Conclusão

O grupo considerou que esta segunda fase do projeto foi demorada, trabalhosa e, em bastantes casos, contraproducente, uma vez que os diagramas de sequência são uma ilusão de auxílio para a compreensão do programa. Devido às numerosas iterações em todo o dinamismo do sistema é apenas natural que a sua tradução para estes diagramas acabe por ser *time consuming* e bastante sensível a qualquer alteração realizada, por exemplo, no diagrama de classes.

Ainda relativamente a este assunto, achamos que os diagramas de sequência seriam mais úteis caso fossem feitos não com os métodos mas através de uma breve descrição das interações entre classes. Neste sentido, somos agora capazes de perceber que os diagramas de sequência não são os mais adequados para todos os cenários visto que, por vezes, a implementação por código fica mais legível que os diagramas, particularmente quando são feitos métodos mais extensos com algoritmos complexos.

Em contraste, o diagrama de classes aparenta ser útil e importante, já que pode ser utilizado como uma espécie de documentação, apresentando os métodos disponibilizados pelas diversas classes. Isto facilita e acelera na produção de resultados.

Por fim, achamos que os diagramas de componentes, de classes e de sequência seguem os requisitos do enunciado, possibilitando a continuação do projeto para a última fase.

Fase III.

Implementação da Solução

13 | Introdução

Este relatório foi realizado no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software, envolvendo a linguagem Java em conjunto com SQL para gestão de base de dados. Tem como objetivo principal ilustrar o comportamento do Sistema de Gestão de Stocks de um armazém de paletes - que irá ser abreviado apenas para Sistema - e explicar o seu funcionamento e o seu fundamento.

O relatório divide-se essencialmente em 4 partes: Na primeira parte, apresentam-se as alterações desde a realização da fase anterior. Já na segunda parte, elabora-se um novo diagrama de classes suportando a arquitetura com DAO. Uma vez que na implementação física concretizou-se uma base de dados, que está sincronizada com o nosso programa, algumas das classes foram modificadas para que quaisquer antigos registo na memória fossem, agora, registados na base de dados. A seguir, faz-se uma análise global ao projeto, desde o inicio até ao fim, através de um exemplo. Para concluir, são referidas algumas considerações gerais acerca do trabalho realizado.

14 | Correções na Lógica de Negócios

14.1. Ator: Gestor

14.1.1. Use Case: Obter listagem de Paletes

Use Case: Obter Listagem de Paletes				
Fluxo Normal	1 Gestor pede listagem ao sistema.		+ listagemPaletes()	TextUI
	2 Sistema verifica paletes existentes, agrupa paletes por zona e calcula o número de prateleiras livres em cada zona.	procura, caso existam, as paletes no armazém	+ listagemPaletes() : String	SSArmazenamento
	3 Sistema mostra ao gestor a listagem de paletes.	sistema imprime no ecrã a lista de paletes, agrupadas por zona (corredor)	+ listagemPaletes()	TextUI

14.2. Ator: Leitor

14.2.1. Use Case: Registrar Paleta

Use Case: Registrar Paleta				
Fluxo Normal	1 Leitor comunica com o sistema as informações do QR-code sobre a paleta	verificar se todas as informações sobre a paleta estão correctas	-validaInfoPaleta(codigoQR : QRCode) : Boolean	SSArmazenamento
	2 Sistema valida todas as informações sobre a paleta			
	3 Sistema determina o tipo de matéria-prima da paleta			
	4 Sistema fica com um registo da paleta	registrar uma paleta no sistema	+ registrarPaleta(codigoQR : QRCode) : String	SSArmazenamento
	5 Sistema adiciona a paleta à lista de paletes a entregar	adicionar paleta registada na zona de descargas	+ adicionaPaletaEntregar(idPaleta : String)	SSArmazenamento
	Fluxo de Exceção 1 2.1 Sistema sinaliza o leitor que as informações não são válidas.	este passo é realizado a partir da obtenção do valor de retorno do método +validaInfoPaleta	-validaInfoPaleta(codigoQR : QRCode) : Boolean	SSArmazenamento

14.3. Ator: Robot

14.3.1. Use Case: Notificar Robot para Transportar (descargas)

Use Case: Notificar Robot para transportar (descargas)				
Fluxo Normal	1 Sistema retira paleta da fila de paletes a entregar	remover paleta da lista de paletes a entregar	+ removePaletaEntregar(idP : String)	SSArmazenamento
	2 Sistema procura prateleira para colocar paleta	procurar uma prateleira que ainda tenha espaço para uma paleta, desde que esta caiba	+ procuraPrateleiraLivre(idPaleta : String) : Prateleira	SSArmazenamento
	3 Sistema procura robot para transportar paleta	procurar um robot para transportar a paleta para a prateleira	+ procuraRobot(p : Paleta) : Robot	SSControloRobot
	4 Sistema calcula rotas para o robot	procurar um robot para transportar a paleta para a zona de entrega	+ calculaRota(r : Robot, posPaleta : Posicao, posDestino : Posicao) : Rota	SSControloRobot
	5 Sistema comunica rotas e paleta ao robot	atualizar o estado do robot de livre para ocupado	+ mudaEstadoRobot(r : Robot)	SSControloRobot
	Fluxo de Exceção 1 2.1 Sistema volta a inserir a paleta na fila de paletes a entregar	adicionar paleta na lista de paletes requisitadas	+ adicionaPaletaEntregar(idP : String)	SSArmazenamento

14.3.2. Use Case: Notificar Recolha

Use Case: Informar Recolha				
Fluxo Normal	1 Robot informa o sistema da recolha da paleta	Informar que robot recolheu a paleta	+ informarRecolha(idRobot : String)	SSControloRobot
	2 Sistema atualiza localização da paleta	Alteração da Posição da Paleta	+ mudaPaletaRobot(idPaleta : String)	SSArmazenamento

14.3.3. Use Case: Notificar Entrega

Use Case: Informar Entrega				
Fluxo Normal	1 Robot informa o sistema da entrega da paleta	Informar que robot entregou a paleta	+ informarEntrega(r : Robot, idPrateleira : String)	SSControloRobot
	2 Sistema atualiza a localização da paleta	Alteração do Local da Paleta	+ mudaPaletaPrateleira(paleta : Paleta, local : Local, idPrateleira : String)	SSArmazenamento
	3 Robot atualiza o seu estado para livre	Alteração do Estado do Robot	+ mudaEstadoRobot(robot : Robot)	SSControloRobot

Nota: O grupo esqueceu-se de expor a transação do use case de obter listagem de paletes, na fase anterior.

15 | Novo Diagrama de Classes

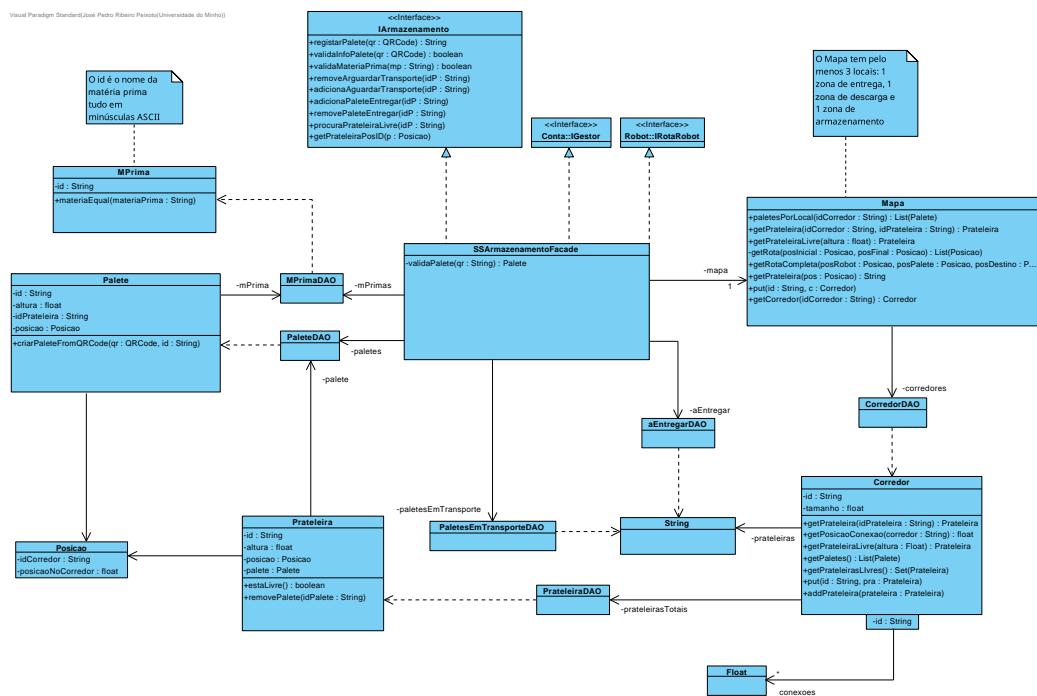
15.1. Abordagem

Para que os dados persistissem entre as múltiplas execuções do programa, desenvolveu-se uma Base de Dados. Com efeito, surgiu a necessidade de alterar o diagrama de classes para estar coerente e compatível com a nova arquitetura.

Além disso, nesta última fase, reduziram-se alguns dos diagramas devido à simplificação do caso de estudo. Mais, adicionaram-se ainda outras classes aos diagramas de casos que não tinham sido considerados. Assim, surgem os seguintes diagramas com a utilização de DAOs.

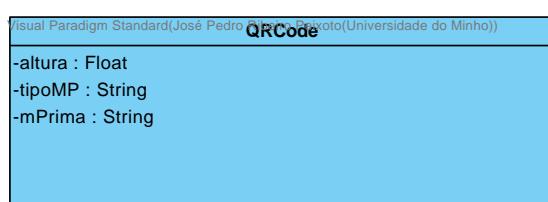
15.2. SSArmazenamentoFacade

15.2.1. Diagrama de Classes



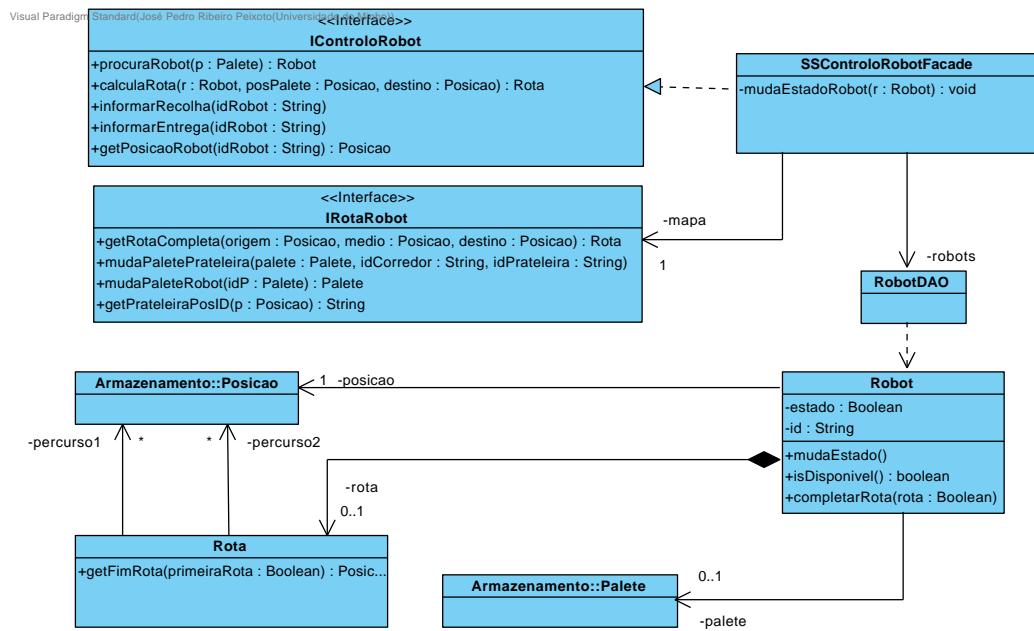
15.3. QRCode

15.3.1. Diagrama de Classes



15.4. SSControloRobotFacade

15.4.1. Diagrama de Classes



15.5. IGestor

15.5.1. Diagrama de Classes



16 | Base de Dados

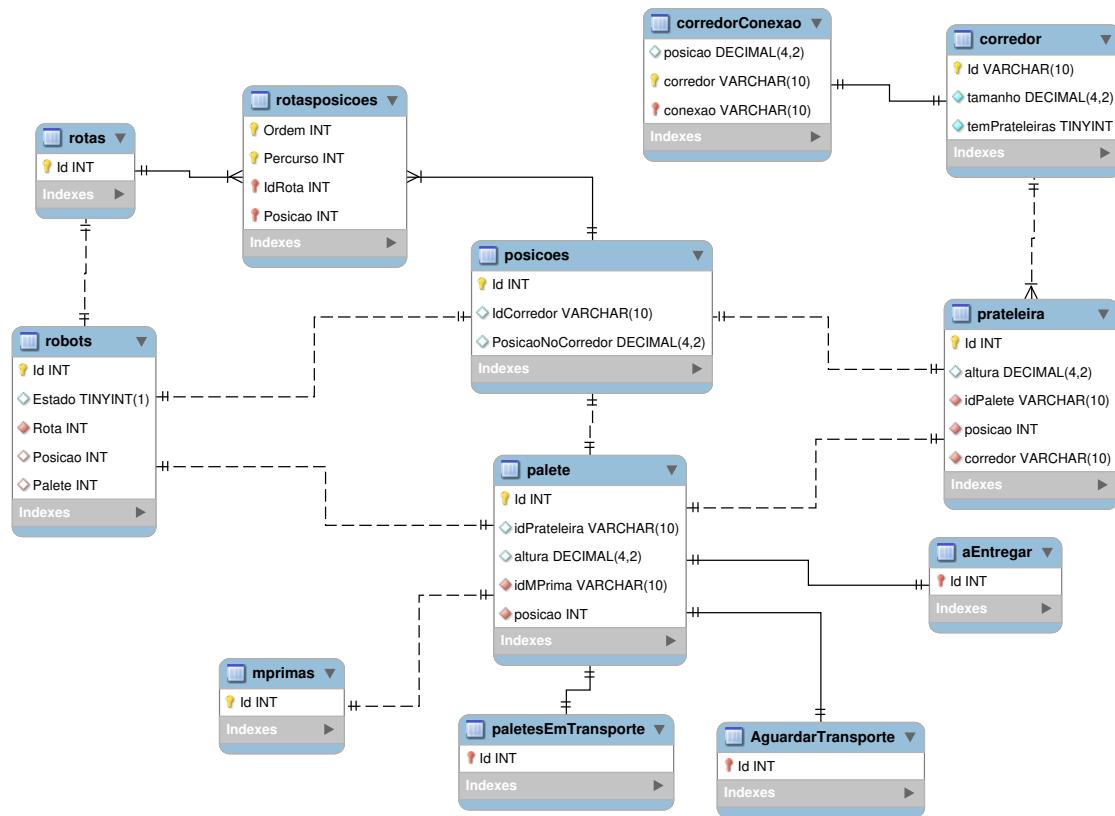
Fundamentação

Para esta fase do projeto, além de uma simples implementação física, o grupo concretizou uma base de dados utilizando como motor o MYSQL. Como será apresentado mais à frente, esta base de dados guarda todas informações essenciais para um correto funcionamento do programa. Desta forma, garantimos que por exemplo, se o programa parar (devido a possíveis erros), todas as informações ficam salva-guardadas. Mais, assim que o programa reinicie, este pode continuar o seu normal funcionamento devido ao acesso a todos os dados que estão na base de dados, mesmo os do momento em que o programa parou.

16.1. Modelo Lógico

Considerando a arquitetura de DAO's do diagrama de classes e, uma vez que um dos objetivos desta fase é desenvolver uma base de dados, foi fundamental definir um modelo lógico. Neste representam-se as tabelas necessárias à gestão dos dados do nosso programa, respeitando-se as regras de mapeamento.

16.1.1. Diagrama



17 | Análise Global ao Projeto

Para a análise global ao projeto, exemplificar-se-á com auxílio de dois use cases - **informar recolha** e **informar entrega** - associados a um robot que transporta paletes.

17.1. Fase 1 : Análise de Requisitos

Contextualização

Nesta fase, analisámos a dinâmica do armazém que indica que os robots transportam paletes entre as várias zonas que lá existem. Sendo assim, sempre que o sistema comunica uma ordem de transporte de paletes, a um robot, este terá de ao longo do percurso notificar da recolha e da entrega da palete, bem como de mudar o seu estado para ocupado ou livre.

Use cases

Sendo assim, o (ator) robot terá associado 3 use cases: comunicar ordem de transporte, notificar entrega, notificar recolha. A identificação dos use cases é essencial para compreender o comportamento do programa e perceber a API que o programa deve oferecer aos atores e, com efeito, o programa é à priori bem estruturado.

17.2. Fase 2 : Modelação Conceptual da Solução

Lógica de Negócios

Adiante, identificam-se os eventuais métodos que compõem os use cases atrás. Mais uma vez, esta identificação acelerará todo o processo de implementação e estruturação física já que, distribui-se os métodos pelos diversos subsistemas do programa. Estes métodos podem ser ainda auxiliados pelos diagramas de sequência.

17.3. Fase 3 : Implementação da Solução

Implementação física

Finalmente, todo o trabalho conseguido nas fases anteriores servirá para engenhar uma solução física, neste caso em Java, para que o ator (ou atores) possa aceder de forma correta e prática à API que lhe foi prometida desde o início (no levantamento de requisitos).

Informar recolha

Neste caso, após o sistema atribuir uma palete (à espera de transporte) a um robot, este deve notificar o sistema assim que a recolha.

```
*** Menu ***
1 - Informar recolha
2 - Informar entrega
0 - Sair
Opção: 1
0 ID do robot
1
```

Informar entrega

Após efetuar a travessia, desde a posição da recolha da palete até à posição (prateleira) onde a palete ficará, este robot deve notificar o sistema assim que entregue a palete.

18 | Conclusão

O grupo considerou que esta terceira e última fase do projeto foi rápida, estimulante e interessante já que concretizou fisicamente os modelos e diagramas anteriormente elaborados.

Antes de mais, referimos a importância da implementação de uma base de dados para o sistema, que numa situação real seria uma das vertentes essenciais no programa. Por isso, foi do nosso maior contentamento ter a oportunidade de aprender e conseguir construir uma conexão entre Java e o MYSQL para a gestão dessa base de dados. Contudo, há aspetos que podem ser aprimorados: todo o acesso à base de dados, bem como o controlo e manipulação dos dados, são executados por queries criadas dentro do Java. Ora, a boa prática era criar as queries à parte, que seriam invocadas pelo programa - que, em termos simples, é como criar os use cases da base de dados.

Além disso, a fase anterior contribuiu, mas não consideravelmente para esta última fase. Como explicado na fase anterior, os diagramas são bastante sensíveis, isto é, surgem impossibilidades ou obstáculos na implementação física que originam mudanças e/ou adaptações nos diagramas (e vice-versa). Ora, isto é completamente normal, contudo continuamos a considerar que a maior parte destes diagramas não compensam o esforço e o tempo dedicado a elaborá-los. Por exemplo, os métodos expostos na lógica de negócios, acabam por ser formulados duas vezes - uma vez no diagrama de sequências e outra na implementação física - e são praticamente idênticos, por isso, consideramos que alguns dos diagramas são redundantes. Ainda assim, o diagrama de classes demonstrou ser um bom suporte à elaboração do nosso trabalho, já que nele contém a API que todos os elementos do grupo (e não só) poderiam utilizar.

Refere-se ainda que, o grupo empenhou-se na generalização e estruturação dos recursos do sistema para facilitar futuras implementações ou melhorias.

Por fim, achamos que os diagramas de DAOs e o programa (em java) seguem os requisitos do enunciado. Deste modo, consideramos que o grupo realizou todas as metas das fases do projeto com eficácia e rigor, contudo referindo que há espaço para futuras implementações e melhorias.