

# **Desenvolvimento de Sistemas de Software**

## **Fase 2**

### **Grupo 34**

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

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

Outubro 2020



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Alterações nos modelos da Fase 1</b>	<b>4</b>
<b>3</b>	<b>Fase 2</b>	<b>4</b>
3.1	Folha Excel . . . . .	4
3.2	Diagrama de Componentes . . . . .	5
3.3	Diagramas de Classes . . . . .	6
3.3.1	Diagrama de Classes do SubStaff . . . . .	7
3.3.2	Diagrama de Classes do SubStock . . . . .	8
3.3.3	Diagrama de Classes do SubRobo . . . . .	9
3.3.4	Diagrama de Classes do SubRequisicao . . . . .	9
3.3.5	Diagrama de Classes Geral . . . . .	10
3.4	Diagramas de Sequência . . . . .	10
3.4.1	Diagrama de Sequência - <i>atualizarRequisicao</i> . . . . .	11
3.4.2	Diagrama de Sequência - <i>Notificar Descarga</i> . . . . .	12
<b>4</b>	<b>Análise Crítica dos Resultados</b>	<b>12</b>
<b>5</b>	<b>Anexos</b>	<b>13</b>
5.1	Use Cases . . . . .	13
5.1.1	Efetuar requisição de material . . . . .	13
5.1.2	Receber itinerário . . . . .	14
5.1.3	Notificação de descarga . . . . .	14
5.1.4	Notificação de recolha . . . . .	15
5.1.5	Registar QR code de palete . . . . .	15
5.1.6	Solicitar listagem de stock . . . . .	16
5.1.7	Autenticação . . . . .	16
5.1.8	Terminar Sessão . . . . .	17
5.2	Diagramas de Sequência . . . . .	18
5.2.1	armazenarPalete . . . . .	18
5.2.2	atualizaOrdem . . . . .	18
5.2.3	atualizarRequisição . . . . .	19
5.2.4	confirmarRequisicao . . . . .	20
5.2.5	criarRequisicao . . . . .	20
5.2.6	efetuarCarga . . . . .	21
5.2.7	efetuarDescarga . . . . .	21
5.2.8	enviarOrdem . . . . .	22
5.2.9	getEstadoPaleteEmPrateleira . . . . .	23
5.2.10	getPaletesDisponiveis . . . . .	24

5.2.11	getStock . . . . .	24
5.2.12	Login . . . . .	25
5.2.13	Logout . . . . .	25
5.2.14	notifacaoCarga . . . . .	26
5.2.15	notificacaoDescarga . . . . .	26
5.2.16	registarErroPalete . . . . .	27
5.2.17	registarConsulta . . . . .	27
5.2.18	registarErroOrdem . . . . .	27
5.2.19	registarLogin . . . . .	28
5.2.20	registarLogout . . . . .	28
5.2.21	registarPalete . . . . .	29
5.2.22	removerRequisicao . . . . .	29
5.2.23	retirarPalete . . . . .	30
5.2.24	validaImpDigital . . . . .	30
5.2.25	ValidaPassword . . . . .	31
5.2.26	validaUsername . . . . .	31

# 1 Introdução

Nesta segunda fase do trabalho, desenvolvemos a arquitetura conceptual do sistema, de modo a suportar o conjunto de *Use Cases* definidos pela equipa docente após a entrega da fase 1. Assim, iremos apresentar:

- as mudanças que fizemos aos diagramas apresentados na fase anterior;
- um diagrama de componentes;
- um diagrama de alto nível onde estão representados os diferentes subsistemas, packages, interfaces e classes;
- um diagrama de classes geral, organizado por packages tal como diagramas de classes para cada subsistema e package;
- os diagramas de sequência que representam o funcionamento dos métodos necessários à implementação dos use cases.

## 2 Alterações nos modelos da Fase 1

No ínicio desta fase apercebemo-nos que precisávamos de alterar certos aspectos dos modelos da fase 1. Por exemplo, no Modelo de Domínio faltaram-nos entidades como prateleiras, ordem de transporte, e houve outras entidades que nos apercebemos que não eram necessárias ou eram irrelevantes, como, por exemplo, ter entidades próprias para a zona de entrega e de descarga, uma vez que as zonas de armazenamento vão ter um identificador e podem ter paletes.

Em relação ao diagrama de *Use Case* chegámos à conclusão que os *Use Cases* que definimos eram mais que suficientes, tirando o facto de que, por lapso, não colocámos o *use case* de *logout* de um utilizador, que é praticamente uma implicação do *use case* de *login*.

As redefinições dos Use Cases estão nos anexos no fim do relatório.

## 3 Fase 2

### 3.1 Folha Excel

Após a reflexão sobre o nosso trabalho na fase 1, decidímos proceder ao levantamento de responsabilidades do sistema, requisitos da API, e subsistemas necessários, a partir dos *use cases*, com recurso ao Microsoft Excel (baseámo-nos no exercício feito na aula prática do dia 13/11), como representado na figura 1.

A	B	C	D	E	F
		Responsabilidades	API	Subsistema	Comentários
1					
2					
3	5.3.1 Efetuar requisição de material				
4	Use case: Efetuar requisição material				
5	Pré condição: true				
6	Pós condição: O sistema fica com um registo das paletes requisitadas				
7	Descrição: Servidor de produção requisita material				
8	Fluxo normal:				
9	1. Servidor Produção comunica paletes a requisitar	registarRequisição	criarRequisicao([lista_materials: Map<String, int>] String)	SubRequisicoes	
10	2. Sistema verifica disponibilidade de paletes	verificarDisponibilidadeDasPaletes	getPaletesDisponiveis([lista_materials: Map<String, int>] Map<String, int>)	subStock	\devolve uma lista com as paletes que estão disponíveis
11	3. Sistema cria um registo de paletes requisitadas	confirmarEstadoDaRequisição	confirmaRequisicao([codRequisicao: String]) void	SubRequisicoes	
12	Fluxo alternativo 1 [alguma palette não disponível] (passo 2)				
13	2.1 Sistema comunica as paletes que não têm disponibilidade	--	--		
14	2.2 Servidor Produção pede cancelamento paletes sem disponibilidade	--	--		
15	2.3 Sistema cancela paletes sem disponibilidade	registerMudancaDeRequisição	atualizarRequisicao([codRequisicao: String, lista_materials: Map<String, int>] void)	SubRequisicoes	\marca as paletes não disponíveis dessa requisição
16	2.4.1 Sistema cancela requisição	--	--		
17	2.4.2 Servidor Produção cancela a requisição de paletes				
18	Fluxo alternativo 2 [requisição por fases] (passo 2.2)				
19	2.2.1 Sistema confirma requisição total	--	--		
20	2.2.2 Sistema cria registo paletes em falta	marcarPaletesEmFaltaNaRequisição	separarRequisicao([codRequisicao: String] Map<String, int>)	SubRequisicoes	\elimina paletes não disponíveis dessa requisição e devolve
21	2.2.3 Sistema cria registo de paletes a entregar	registerRequisição	criarRequisicaoAdiada([lista_materials: Map<String, int>] String)	SubRequisicoes	\cria uma nova requisição com as paletes em falta
22	Fluxo de excepção 1 [requisição não pode ser fases] (passo 2.2)				
23	2.2.1 Servidor Produção cancela requisição de paletes				
24	Fluxo Normal:				
25	5.3.4 Receber Itinerário				
26	Use Case: Receber Itinerário				
27	Pré condição: True				
28	Pós condição: O sistema fica com registo sobre o estado final do robô				
29	Descrição: O sistema comunica o percurso e tarefas a um robô				
30	Fluxo Normal:				
31	1. O sistema comunica ao robô o seu próximo itinerário de trabalhos	enviarItinerarioAoRobô	enviaOrdem(o: OrdemTransporte) Boolean	SubRobos	\devolve false se estiver ocupado
32	2. O robô confirma a receção das instruções	--	--		
33	3. O sistema regista o estado final do robô	mudarEstadoDoRobô	atualizaEstado([codRobo: String, e: EstadoRobo]) void	SubRobos	
34	Fluxo alternativo 1 [O robô já possui uma lista de tarefas] (passo 2)				
35	2.1 O robô comunica ao sistema o seu estado atual	--	--		
36	2.2 O sistema indica ao robô para prosseguir com o itinerário fornecido	mudarItinerarioDoRobô	atualizaOrdem(o: OrdemTransporte) Boolean	SubRobos	
37	2.3 O robô confirma a mudança de instruções	--	--		
38	2.4 Volta no passo 3				

Figura 1: Excerto da Folha Excel

Nota: Ao longo do desenvolvimento desta fase fizemos alterações aos métodos, pelo que as definições que apresentaremos posteriormente podem não corresponder 100% às desta imagem.

No fim deste processo chegámos à conclusão que poderíamos dividir a nossa arquitetura em 4 subsistemas diferentes:

1. **subRobo** - O subRobo é o subsistema que controla os robôs e as suas ordens de transporte.
2. **subStaff** - O subStaff é o subsistema que controla todas as ações que estão relacionadas com funcionários do armazém, como por exemplo a autenticação de gestores.
3. **subStock** - O subStock é o subsistema que engloba toda a logística do armazém, isto é, localização de paletes (prateleira e zona) e a matéria prima das mesmas.
4. **subRequisicoes** - O subRequisições tem como objetivo controlar todos os processos que estão relacionados com as requisições feitas pelo servidor de produção.

Para além disto, acreditámos que, apesar de trabalhosa, esta fase poupou-nos imenso trabalho no futuro, uma vez que também ficámos com uma lista de métodos que teria de ser suportada na nossa API e as classes envolvidas.

### 3.2 Diagrama de Componentes

Como já tínhamos uma ideia dos subsistemas que a nossa arquitetura iria ter, este diagrama foi rapidamente concluído. Começámos por representar a nossa lógica de negócio

que iria incorporar os nossos 4 subsistemas. De seguida decidimos que cada subsistema teria uma Interface para representar a sua API, tal como uma classe Facade.

O objetivo de usarmos o Padrão de Projeto Facade é encapsular os nossos subsistemas, escondendo a sua complexidade e torná-los mais fáceis de usar, de modo a que, se no futuro precisarmos de fazer mudanças ao subsistema não afetemos o cliente que o utiliza.

Para além disso, teríamos uma classe Facade para a lógica de negócios em geral que decidimos chamar de ArmazemFacade. Por último adicionámos um componente para representar uma ou mais vistas ("ArmazemUI") e um componente para representar a camada de Dados ("ArmazemDL").

Podemos ver o diagrama resultante na figura 2.

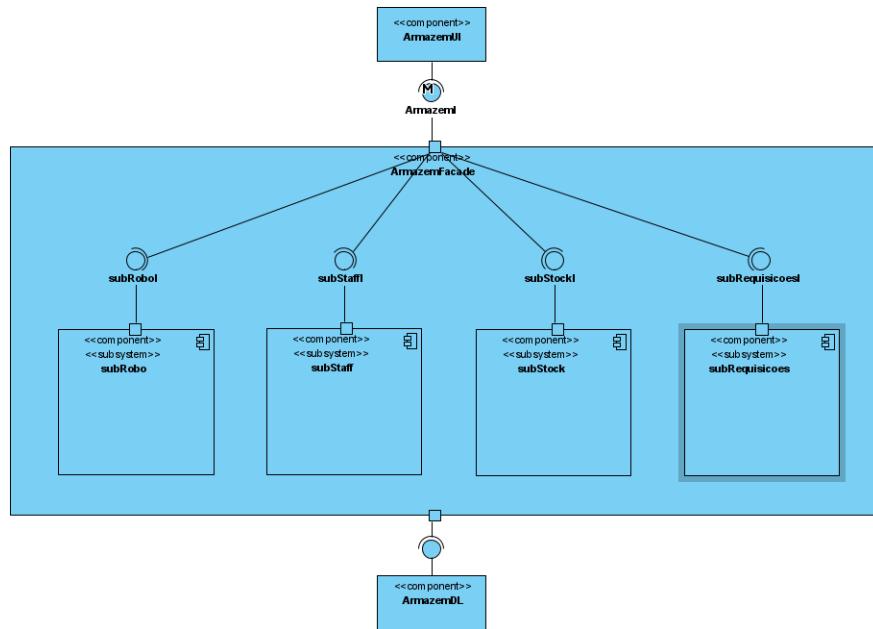


Figura 2: Diagrama de Componentes

### 3.3 Diagramas de Classes

Como temos vários subsistemas do nosso sistema, de modo a organizar melhor o nosso diagrama, decidimos dividir as classes por packages, que, no fundo, representam cada sistema. Assim, temos um diagrama de classes para cada um dos Packages/Subsistema. Para além destes diagramas, decidimos ter um diagrama de classes geral com todos os nossos subsistemas, sistema e interfaces.

### 3.3.1 Diagrama de Classes do SubStaff

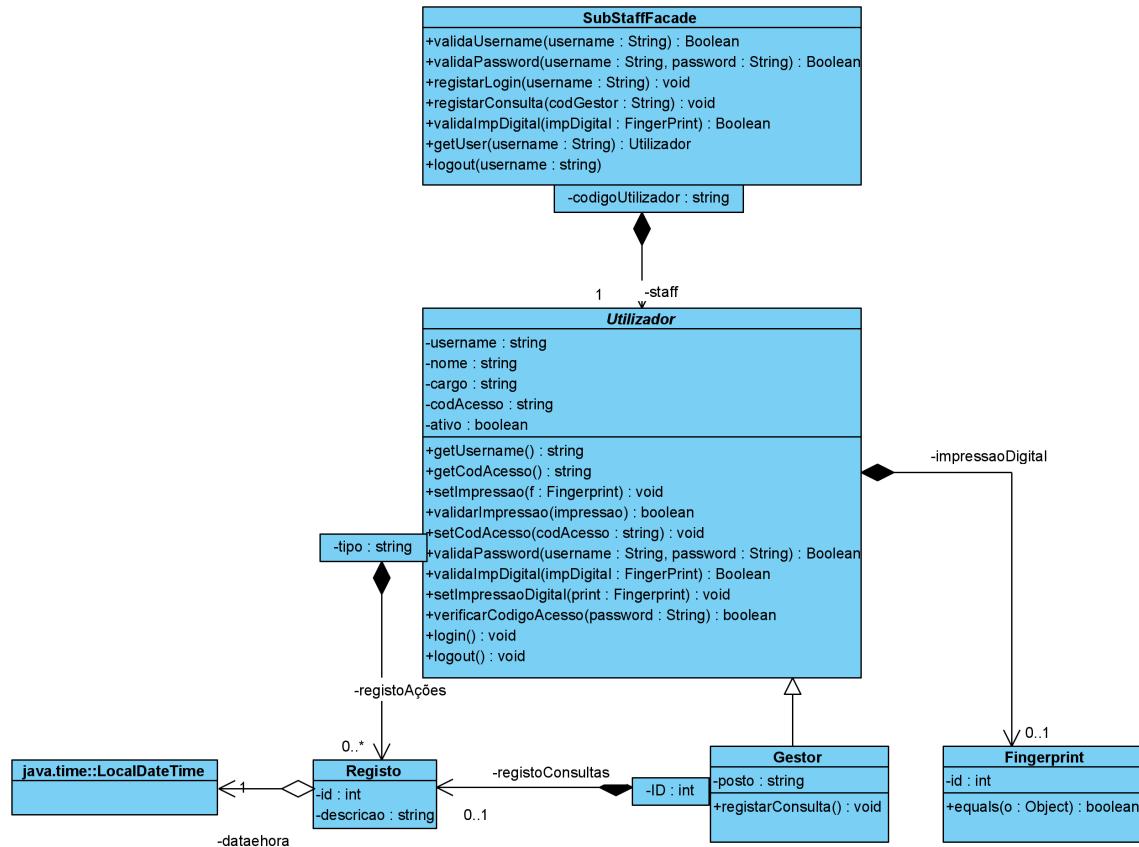


Figura 3: Diagrama da classe do subsistema SubStaff

### 3.3.2 Diagrama de Classes do SubStock

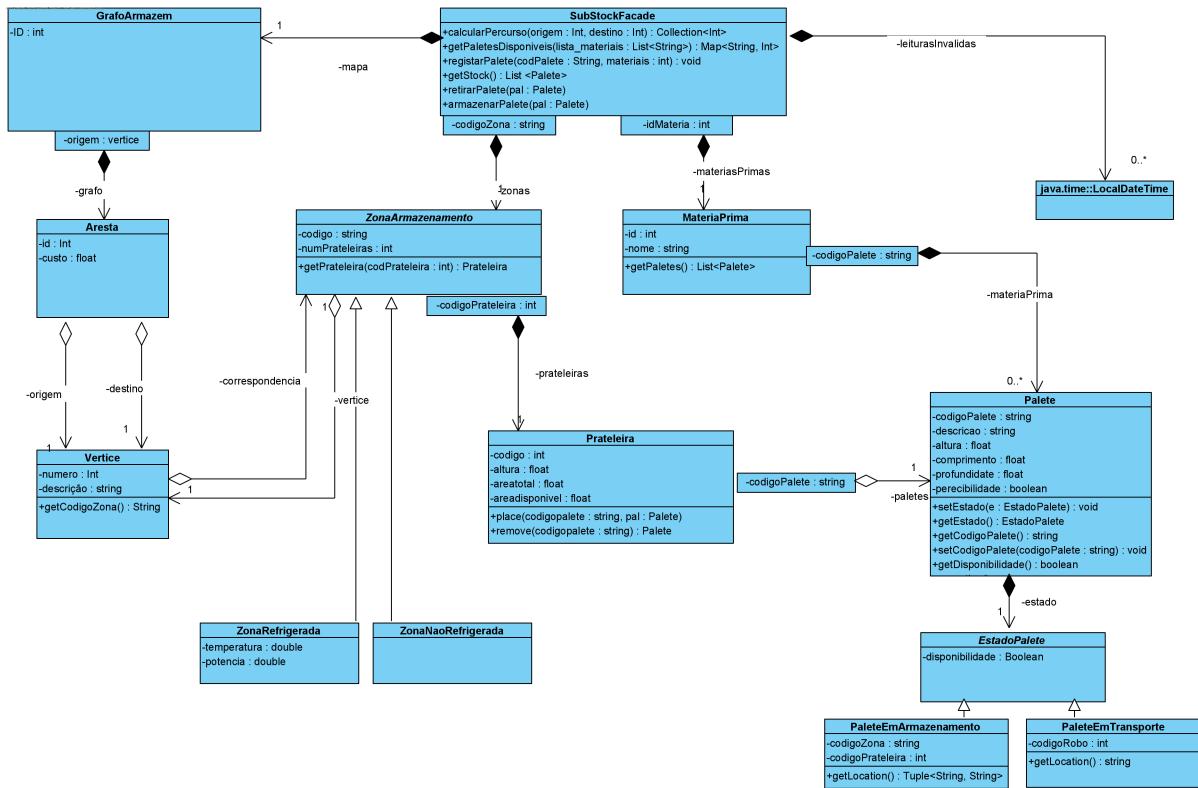


Figura 4: Diagrama da classe do subsistema SubStock

### 3.3.3 Diagrama de Classes do SubRobo

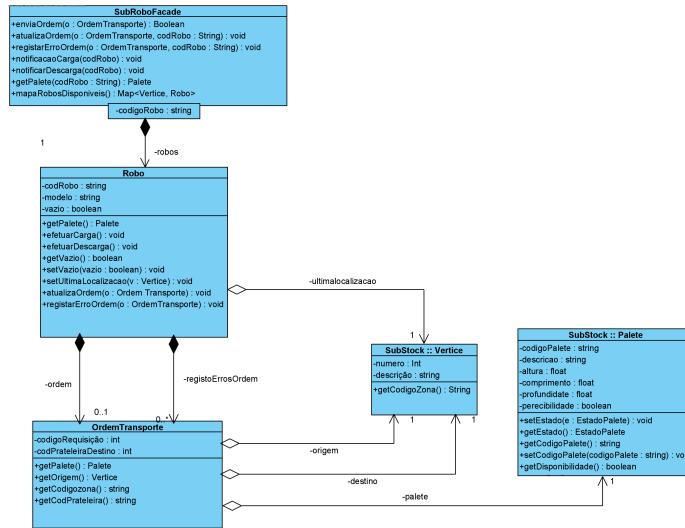


Figura 5: Diagrama da classe do subsistema SubRobo

### 3.3.4 Diagrama de Classes do SubRequisicao

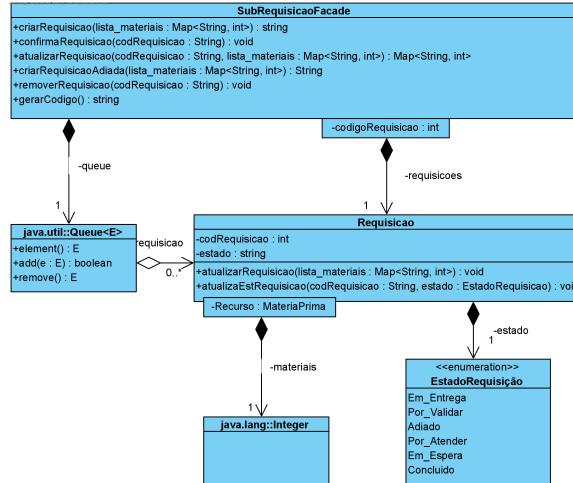


Figura 6: Diagrama da classe do subsistema SubRequisicao

### 3.3.5 Diagrama de Classes Geral

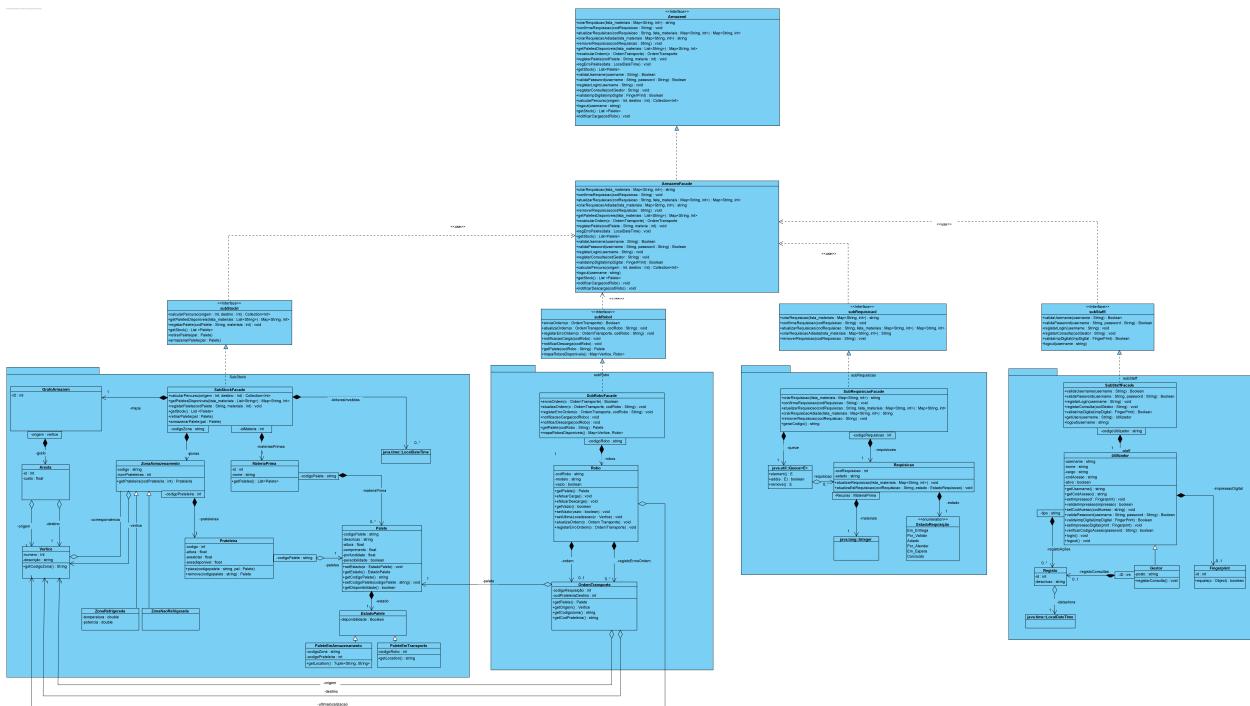
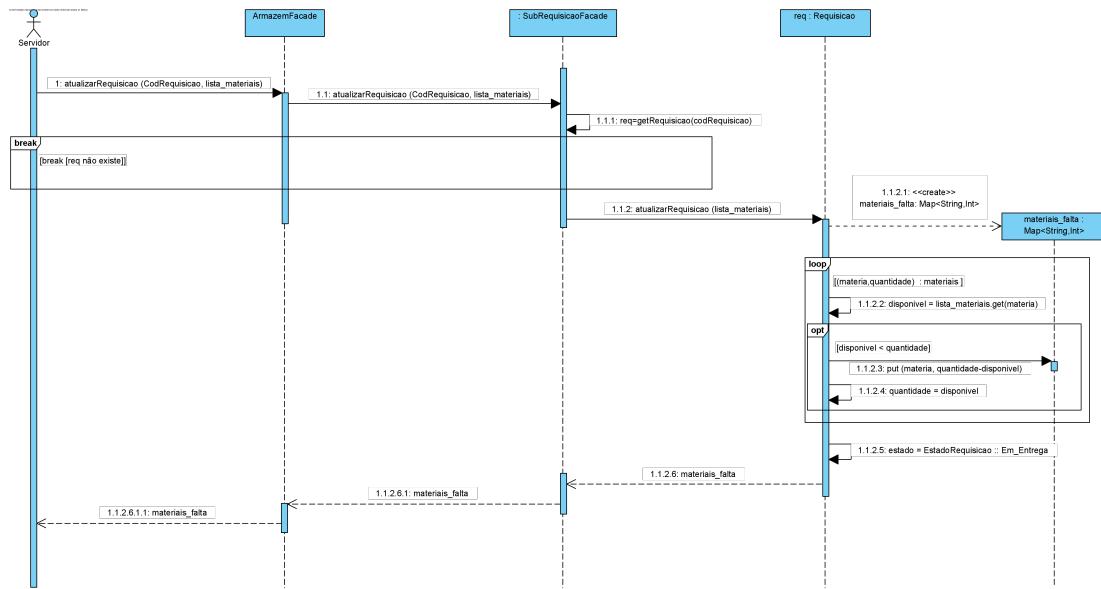


Figura 7: Diagrama de classes geral

### 3.4 Diagramas de Sequência

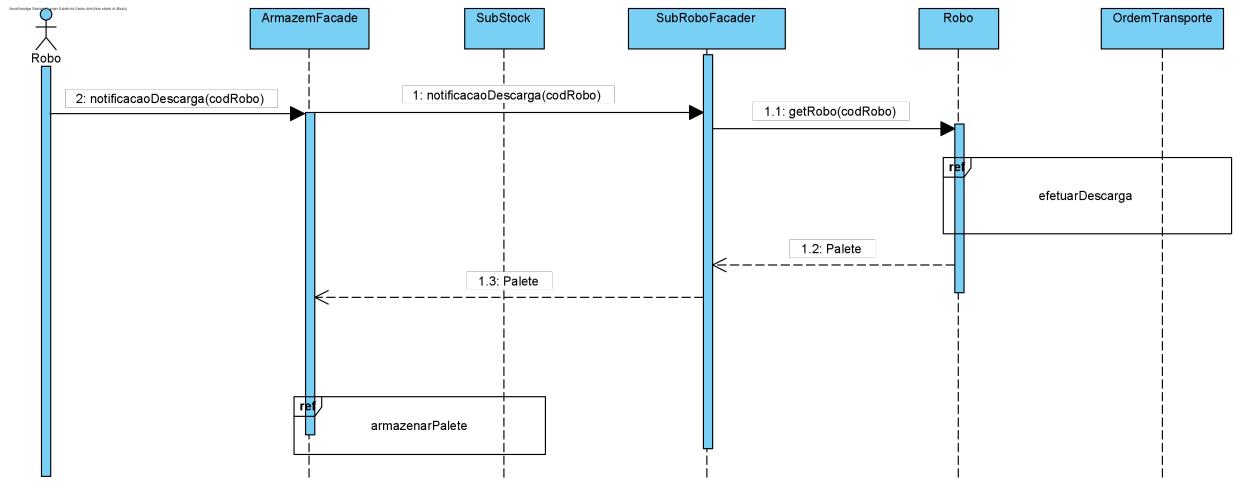
Nesta fase, procedemos à modelação comportamental dos métodos necessários à implementação dos Use Cases e outros que achámos necessários. Para tal efeito fizemos diagramas de sequência para cada método. Como são demasiados para abranger neste relatório iremos apenas abordar os mais interessantes e complexos. No entanto, tanto nos anexos como no ficheiro do projeto .vpp estão presentes todos os diagramas criados.

### 3.4.1 Diagrama de Sequência - *atualizarRequisicao*



O método *atualizarRequisicao* tratado pelo subsistema de Requisições tem como objetivo separar as matérias em falta de uma dada requisição, devolvendo as mesmas numa coleção. Sendo assim, o servidor chama o método na lógica de negócios que por sua vez chama o mesmo método no subsistema correspondente. Após validar o código dado, este localiza o respetivo objeto Requisição e começa uma repetição em que calcula todos os materiais em falta com base na lista de materiais guardados, guardando-os por fim num *Map* que foi previamente criado. Por fim, após todas as matérias serem válidas, este atualiza o estado da Requisição, ativando-a, e termina devolvendo o *Map* de materiais em falta ao Servidor.

### 3.4.2 Diagrama de Sequência - Notificar Descarga



O método *notificarDescarga* tratado pelo subsistema de Robos tem como objetivo notificar o sistema que o robô realizou uma descarga. Desta forma, o robô chama o método na lógica de negócios, que por sua vez chama o mesmo método no subsistema do Robô. Daqui, o subsistema do Robô trata de localizar o robô e este chama o método *efetuarDescarga*, que atualiza o estado interno do robô e devolve a paleta descarregada. Esta será comunicada à lógica de negócios que invocará o método *armazenarPaleta* ao subsistema Stock. Por fim, este encarregar-se-á de atualizar os registos das paletes armazenadas.

## 4 Análise Crítica dos Resultados

Esta fase foi extremamente complexa, uma vez que qualquer pequena alteração que seja feita terá imensas implicações na fase de implementação, implicações estas que eram mais óbvias do que na fase 1, uma vez que estávamos a trabalhar numa “camada” mais próxima desta.

Apesar de ter sido preciso alterar alguns aspetos da fase 1, os diagramas desta fase estão de acordo com as nossas ideias iniciais, o que corrobora a nossa confiança no relatório anterior de que o modelo de domínio e os *use cases* foram bem definidos, e revelaram-se úteis, para esta segunda fase.

Focando-nos agora nesta fase, rapidamente nos apercebemos que esta seria a que mais nos desafiaría, uma vez que tivemos de repensar e mudar os nossos raciocínios de estruturação e de relações entre classes inúmeras vezes. Porém, acreditámos que este nosso esforço será recompensado, na medida em que já resolvemos uma grande parte dos problemas que nos poderiam aparecer durante a fase de implementação.

Em suma, tal como aconteceu com a fase 1, estamos confiantes de que, durante a próxima fase, iremos precisar de fazer poucas alterações, ou idealmente nenhuma, dado que à medida que nos aproximamos na implementação física, quaisquer alterações terão bastante mais implicações e consequências.

## 5 Anexos

### 5.1 Use Cases

1. Gestor
  - i Consultar listagem de localizações
  - ii Iniciar sessão
  - iii Terminar sessão
2. Leitor de códigos QR
  - i Comunicar código QR
3. Robot
  - i Sistema comunica ordem de transporte
  - ii Notificar recolha de paletes
  - iii Notificar entrega de paletes
4. Servidor da produção
  - i Requisitar paletes

#### 5.1.1 Efetuar requisição de material

Use case : Efetuar requisição material

Pré condição: true

Pós condição: O sistema fica com um registo das paletes requisitadas

Descrição: Servidor de produção requisita material

Fluxo normal:

- 1- Servidor Produção comunica quais as paletes a requisitar
- 2- Sistema valida disponibilidade das paletes
- 3- Sistema cria um registo de paletes requisitadas

Fluxo alternativo 1 [alguma palete não disponível] (passo 2)

2.1 Sistema comunica as paletes que não têm disponibilidade

- 2.2 Servidor produção pede cancelamento paletes sem disponibilidade
- 2.3 Sistema cancela paletes sem disponibilidade
- 2.4 Regressa ao passo 3

Fluxo alternativo 2 [requisição por fases] (passo 2.2)

- 2.2.1 Sistema confirma requisição total
- 2.2.2 Sistema cria registo paletes em falta
- 2.2.3 Sistema cria registo de paletes a entregar

Fluxo de exceção 1 [requisição não pode ser por fases] (passo 2.2)

- 2.2.1 Servidor Produção cancela requisição de paletes

### 5.1.2 Receber itinerário

Use Case: Receber Itinerário

Pré condição: True

Pós condição: O sistema fica com registo sobre o estado final do robô

Descrição: O sistema comunica o percurso e tarefas a um robô

Fluxo Normal:

- 1. O sistema comunica ao robô o seu próximo itinerário de trabalhos
- 2. O robô confirma a receção das instruções
- 3. O sistema regista o estado final do robô

Fluxo Alternativo 1 [O robô já possui uma lista de tarefas] (passo 2):

- 2.1 O robô comunica ao sistema o seu estado atual
- 2.2 O sistema indica ao robô para prosseguir com o itinerário fornecido
- 2.3 O robô confirma a mudança de instruções
- 2.4 Volta ao passo 3

### 5.1.3 Notificação de descarga

Use case : Notificação de descarga

Pré condição: Robo está a transportar uma paleta

Pós condição: Sistema fica com o registo do sucesso da descarga

Descrição: Robo informa ao sistema sobre uma tentativa de descarga

Fluxo Normal:

1. Robo informa ao sistema que efetuou uma descarga
2. Sistema regista o estado novo do robo e o sucesso da operação

Fluxo Excepcional [Local de descarga esta indisponivel](passo 1):

- 1.1 Robo informa que nao consegue descarregar a palete naquele local
- 1.2 Sistema regista erro na descarga
- 1.3 Sistema calcula um procedimento alternativo
- 1.4 Sistema comunica ao robo o novo procedimento
- 1.5 Robo comunica receção do procedimento alternativo

#### **5.1.4 Notificação de recolha**

Use case : Notificação de recolha

Pré condição: Robo não está a transportar uma palete

Pós condição: Sistema fica com o registo do sucesso da recolha

Descrição: Robo informa ao sistema sobre uma tentativa de recolha

Fluxo Normal:

1. Robo informa ao sistema que efetuou uma recolha de uma palete
2. Sistema regista o estado novo do robo e o sucesso da operação

Fluxo Excepcional 1 [Nao existe nenhuma palete identificada a ser recolhida] (passo 1):

- 1.1 Robo informa ao sistema que não existe nada para recolher no sitio onde encontra
- 1.2 Sistema regista o erro na recolha
- 1.3 Sistema regista o novo estado do robô

#### **5.1.5 Registar QR code de palete**

Use case: Registar QR code de palete

Pré condição: True

Pós condição: O sistema fica com um registo de uma nova palete

Descrição: O scanner lê e comunica ao sistema o código de uma paleta descarregada

Fluxo Normal:

- 1 Scanner envia codigo da palete lido

2. O sistema regista uma nova palete

Fluxo de Exceção 1 [Leitura Invalida] (passo 1):

1.1 Scanner informa o erro ao sistema

1.2 Sistema regista o erro

#### **5.1.6 Solicitar listagem de stock**

Use case: Solicitar listagem de stock

Pré condição: True

Pós condição: Listagem de todo o stock disponível

Descrição: Gestor solicita listagem de stock ao sistema

Fluxo Normal:

1. Gestor solicita listagem de stock ao sistema
2. Sistema gera listagem de stock
3. Sistema apresenta a listagem de stock
4. Sistema regista consulta de listagem de stock

#### **5.1.7 Autenticação**

Use case: Autenticação

Pré-condição: True

Pós-condição: O ator fica autenticado

Descrição: O ator autentica-se ao sistema

Fluxo Normal:

1. O sistema pede ao ator o seu nome de utilizador
2. O ator fornece o seu nome de utilizador
3. O Sistema valida o nome de utilizador
4. O Sistema pede ao ator o seu código de acesso
5. O ator fornece o seu código de acesso
6. O sistema valida o código de acesso
7. O sistema informa o ator que autenticou-se com sucesso
8. O sistema regista a autenticação do ator

Fluxo Excepcional 1 [O nome de utilizador é invalido](passo 3):

3.1. O sistema informa o ator que o nome de utilizador não é valido

Fluxo Excecial 2 [O código de acesso é invalido](passo 6):

6.1. O sistema informa o ator que o código de acesso não é valido

Fluxo Alternativo 1 [O ator autentica-se por impressão digital](passo 1)

1.1. O ator apresenta o seu dedo ao sistema

1.2. O sistema valida a impressão digital

1.3. Volta ao passo 7

Fluxo Excecial 3 [A Impressão digital não é reconhecida](passo 1.2)

1.2.1. O sistema informa o ator que a impressão digital não é reconhecida

### **5.1.8 Terminar Sessão**

Use case: Terminar sessão

Pré-condição: Ator estar loggado no sistema

Pós-condição: Ator termina sessão e fica registado no sistema o fecho de sessão

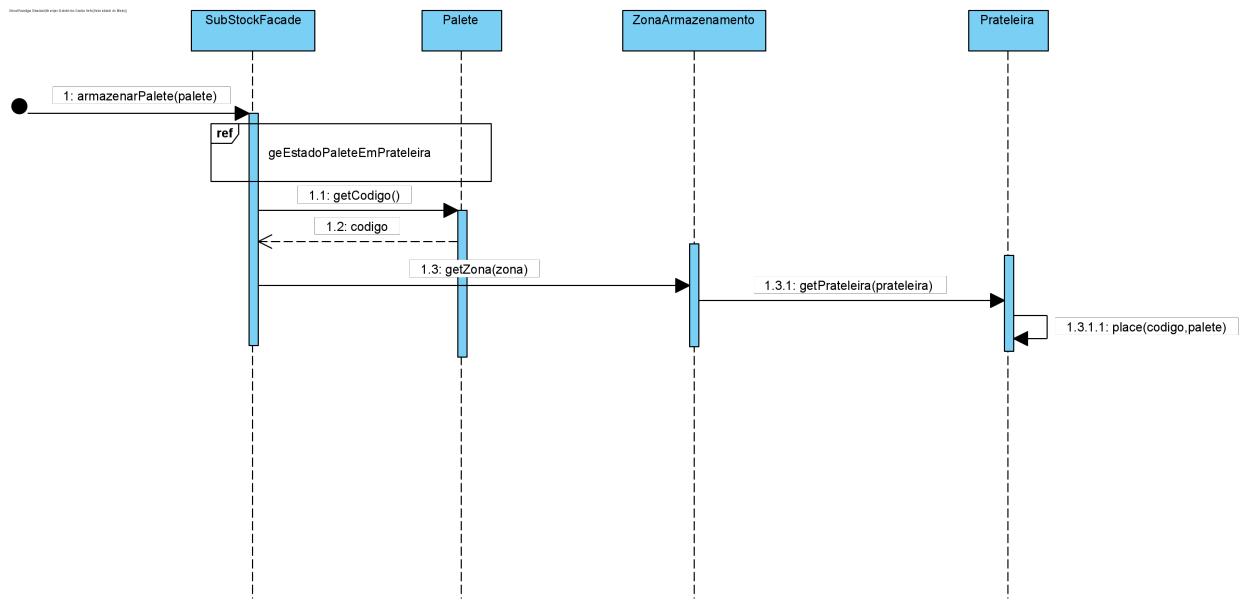
Descrição: Término de sessão por parte do ator

Fluxo Normal

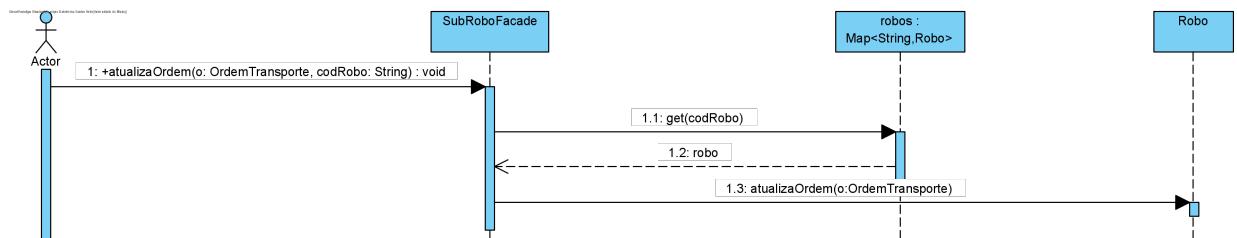
1. Ator termina sessão
2. Sistema regista logout do ator

## 5.2 Diagramas de Sequência

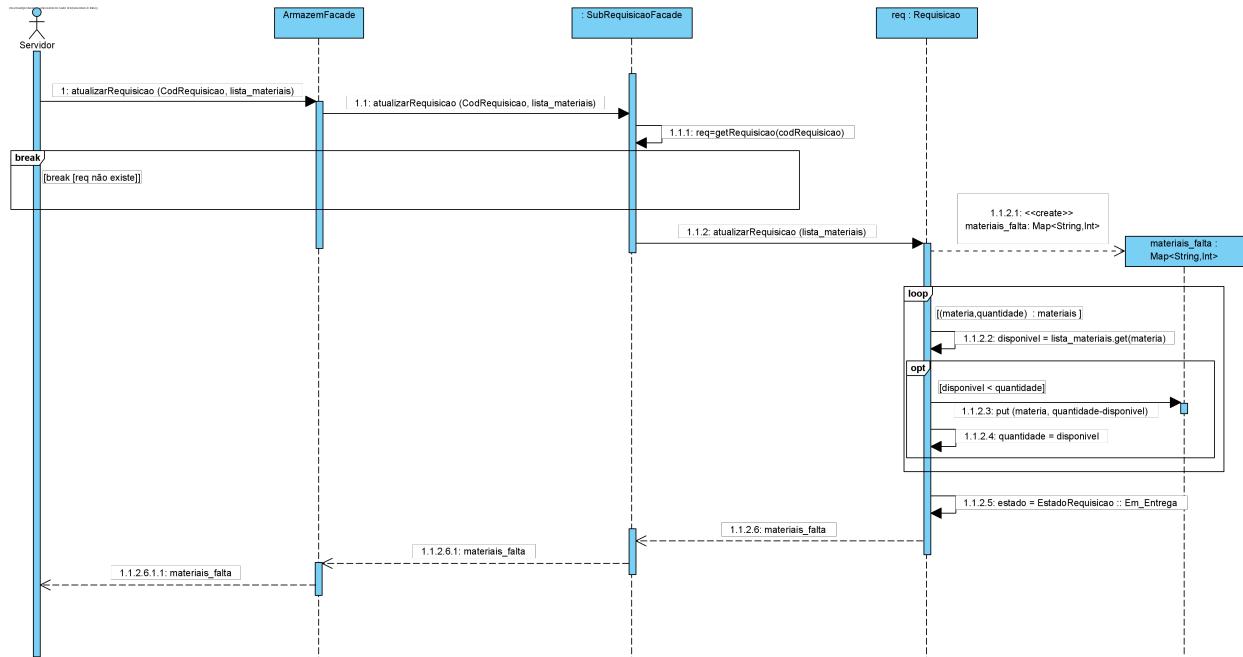
### 5.2.1 armazenarPalete



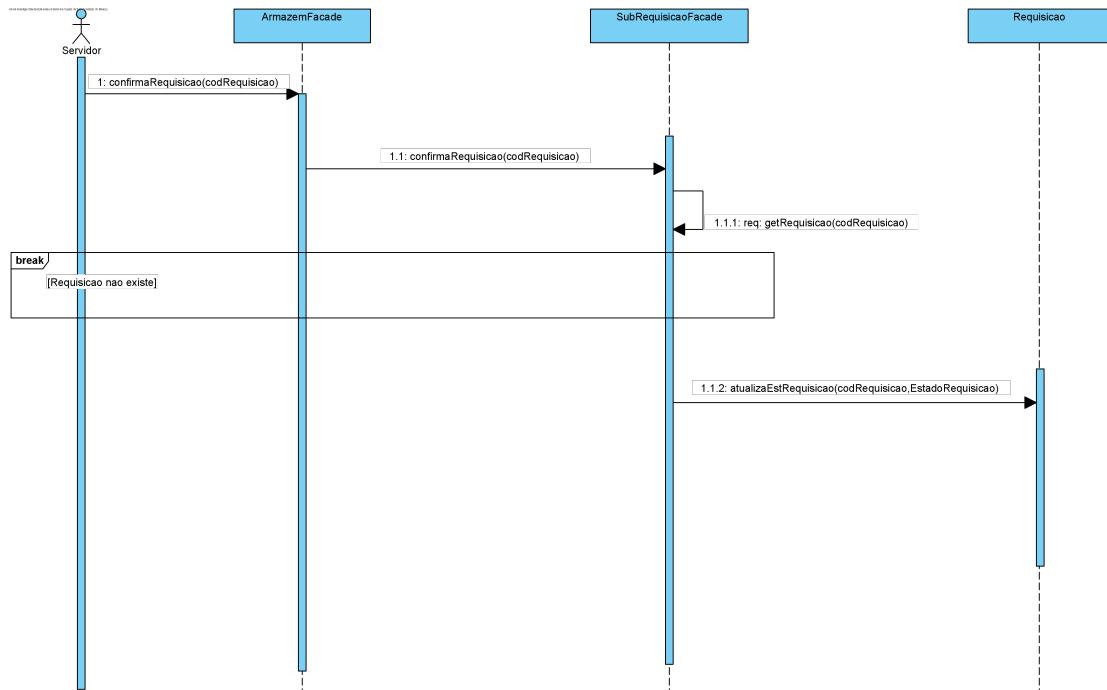
### 5.2.2 atualizaOrdem



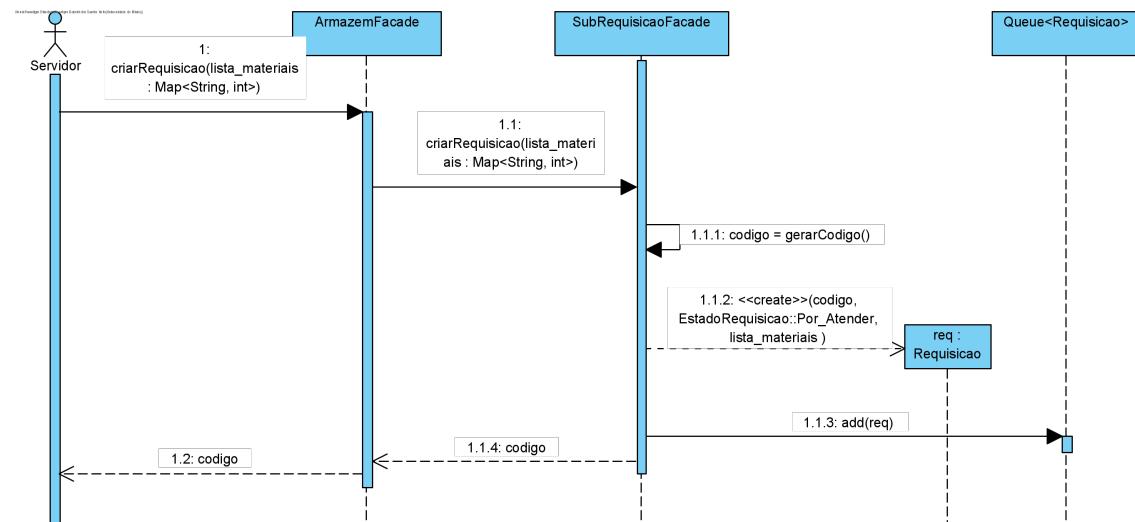
### 5.2.3 atualizarRequisição



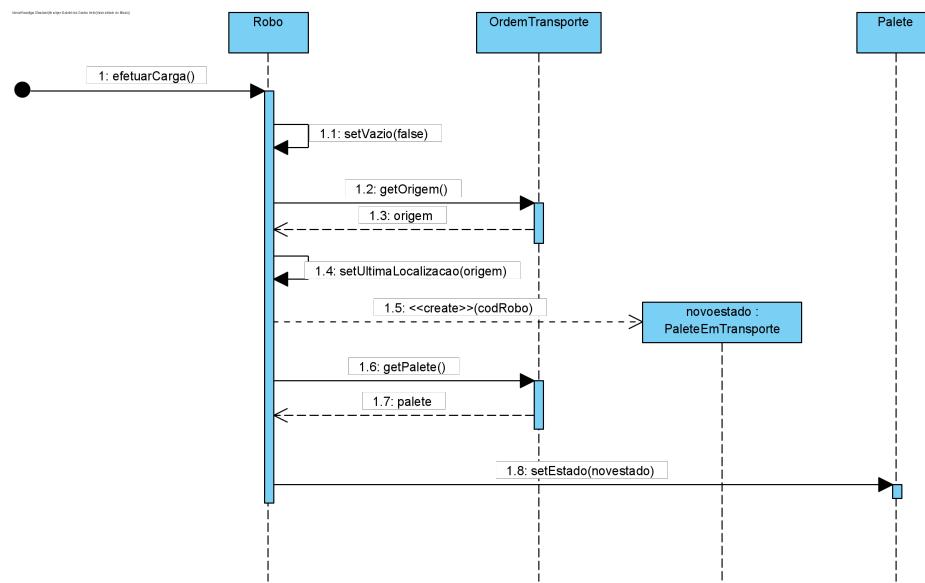
### 5.2.4 confirmarRequisicao



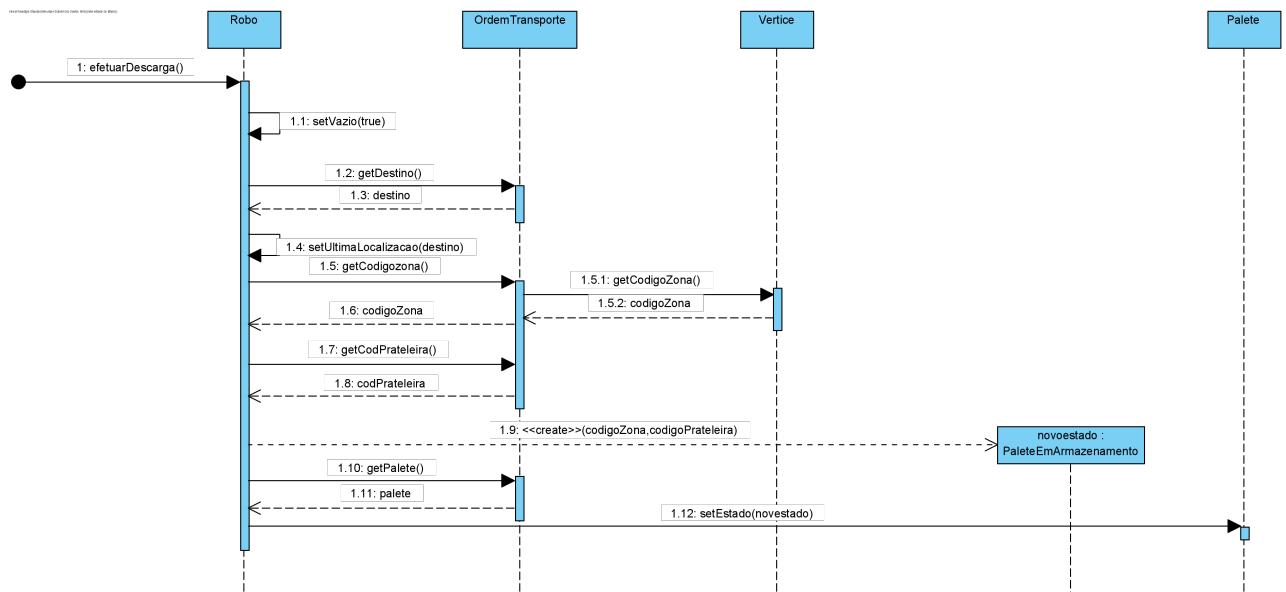
### 5.2.5 criarRequisicao



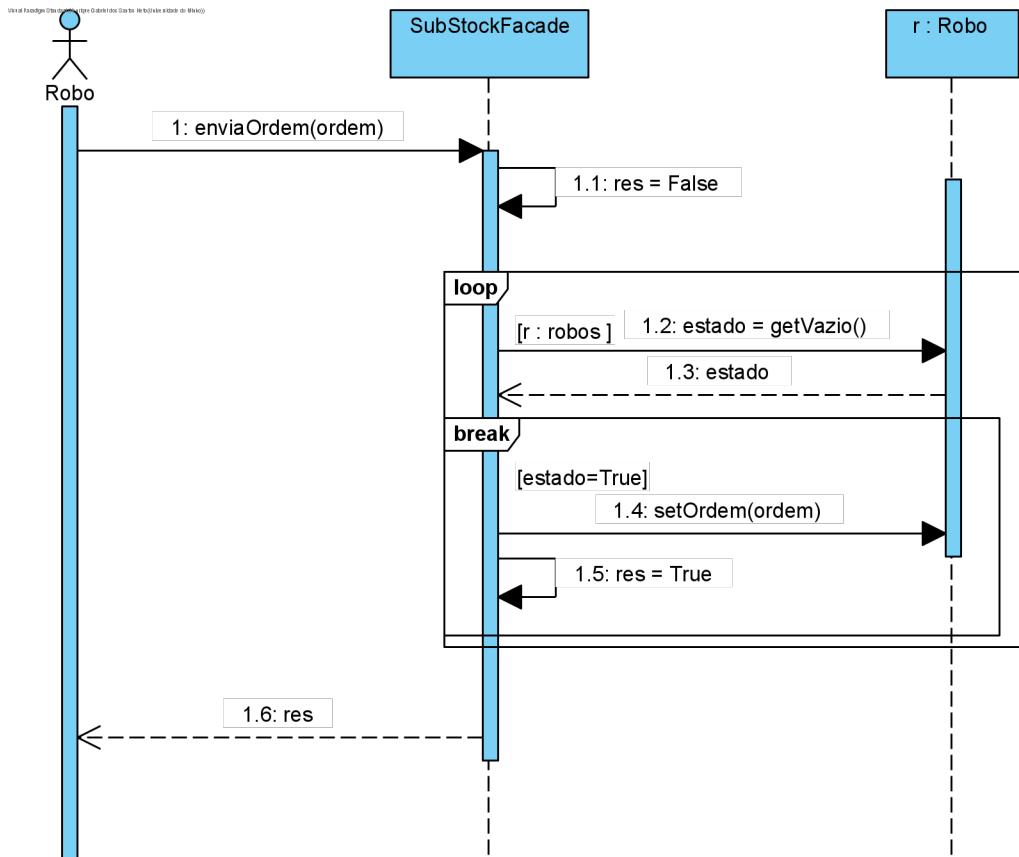
### 5.2.6 efetuarCarga



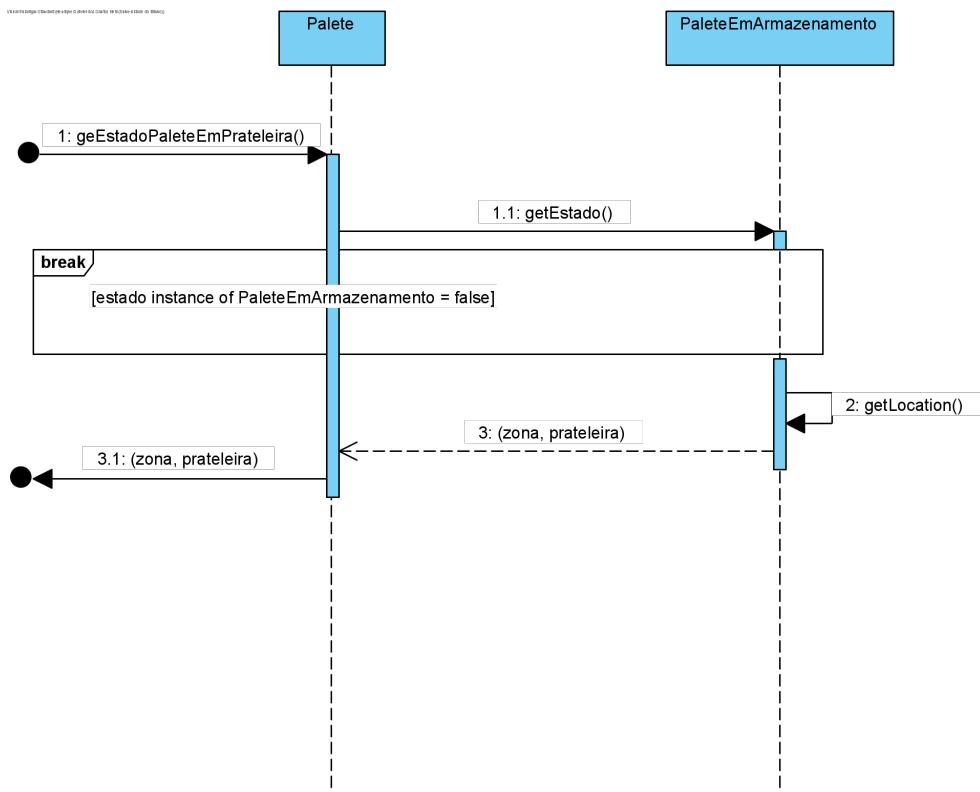
### 5.2.7 efetuarDescarga



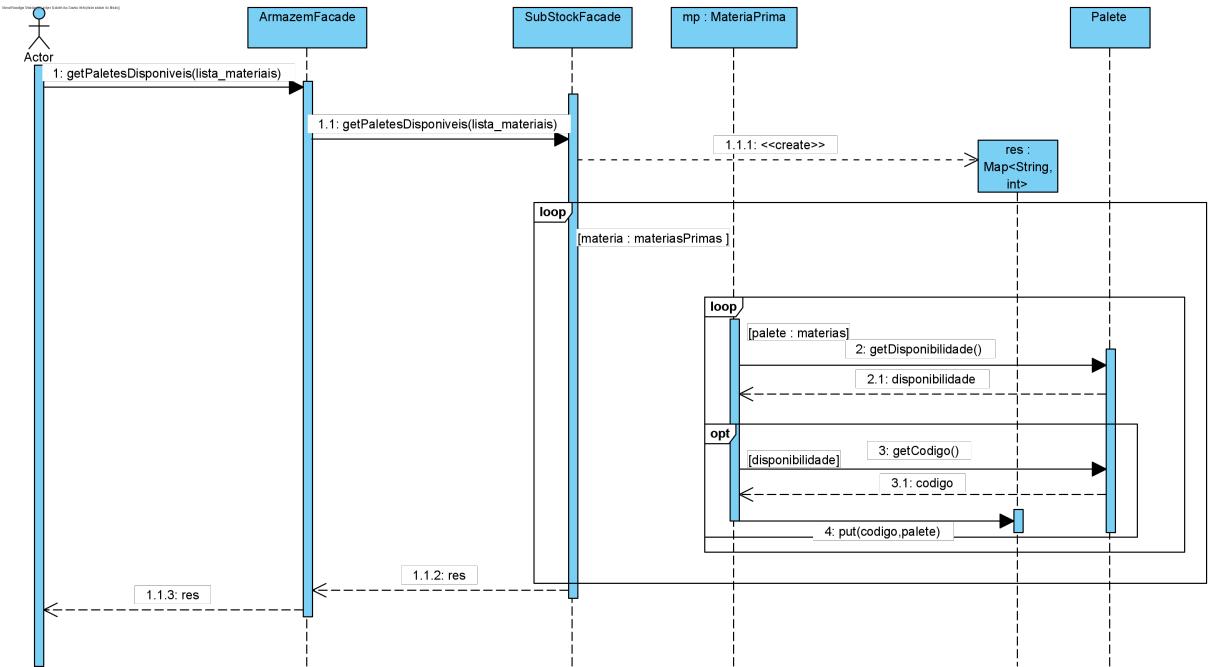
### 5.2.8 enviarOrdem



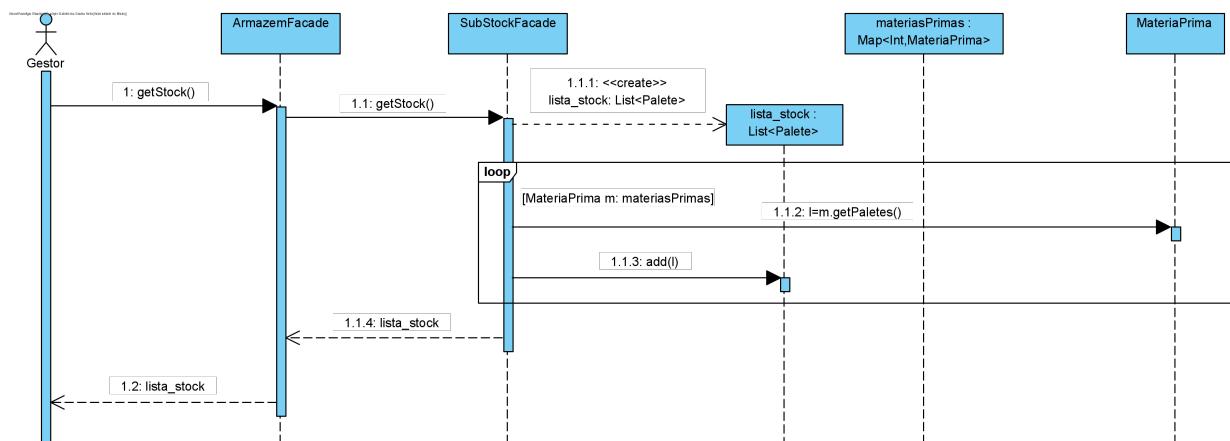
### 5.2.9 getEstadoPaleteEmPrateleira



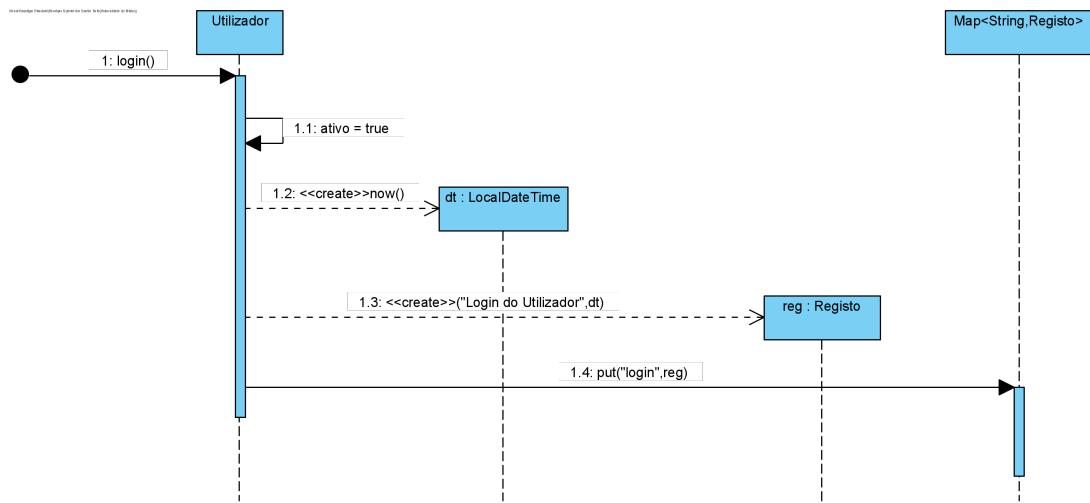
### 5.2.10 getPaletesDisponiveis



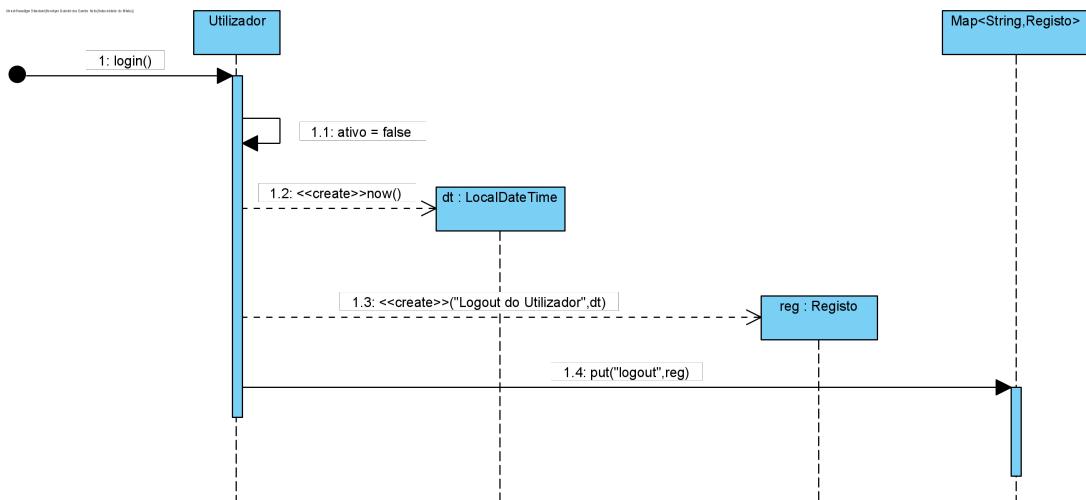
### 5.2.11 getStock



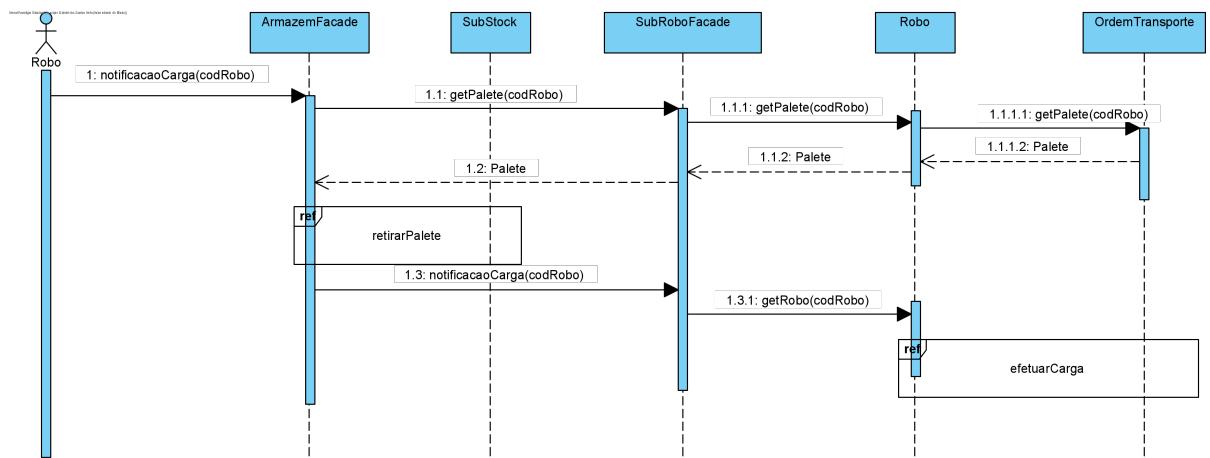
### 5.2.12 Login



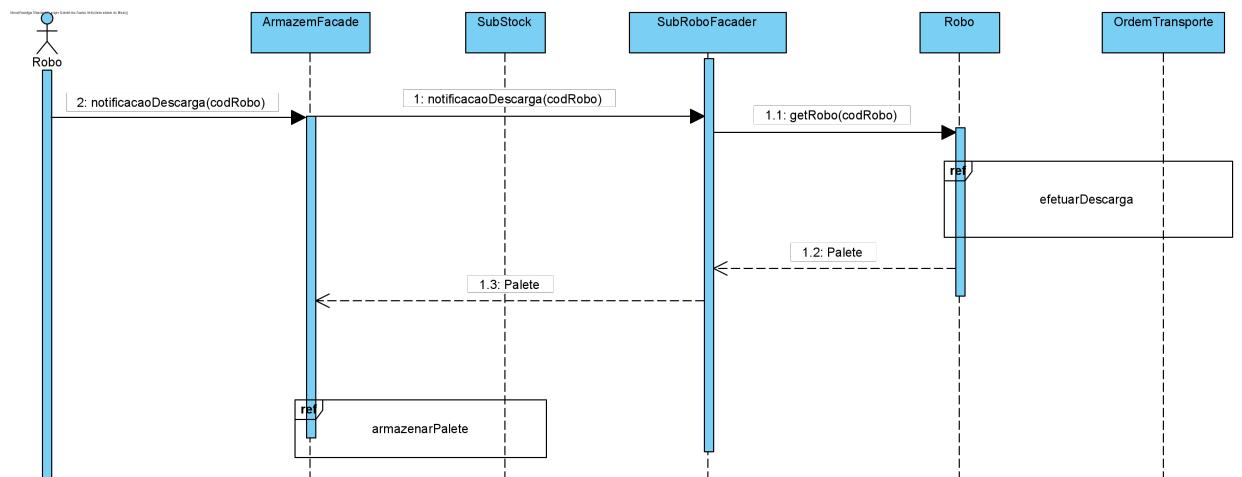
### 5.2.13 Logout



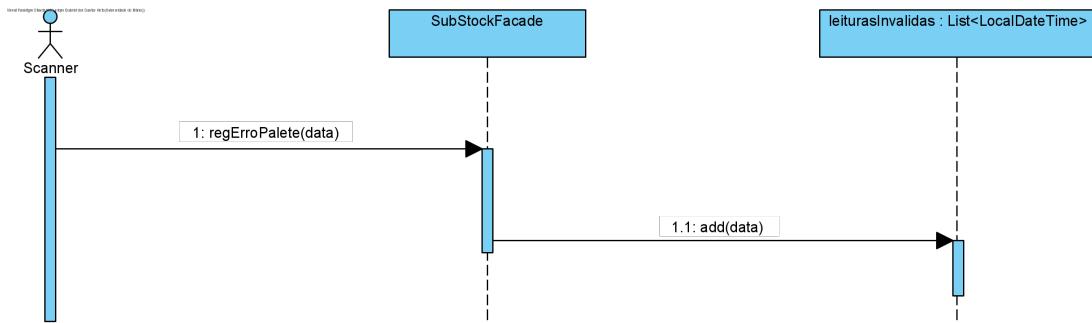
### 5.2.14 notificacaoCarga



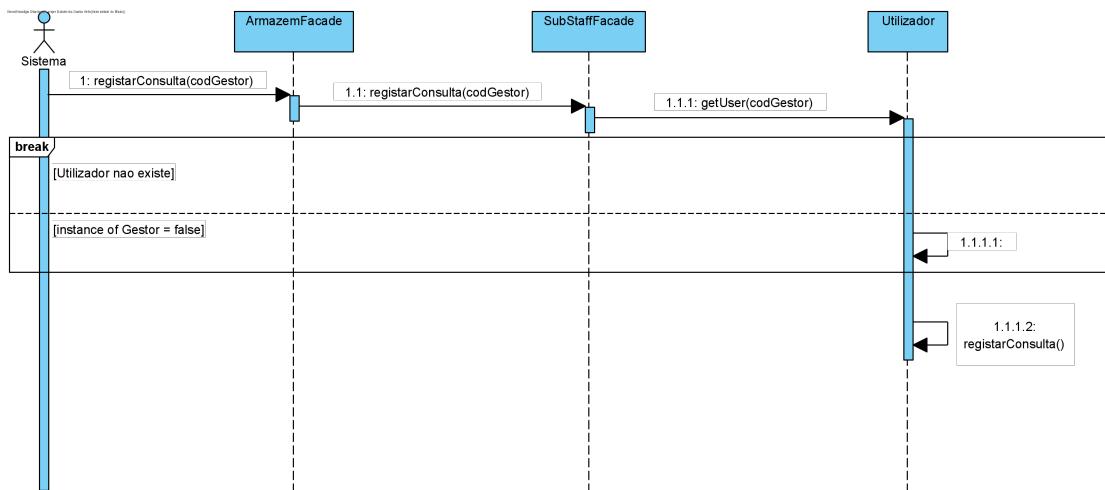
### 5.2.15 notificacaoDescarga



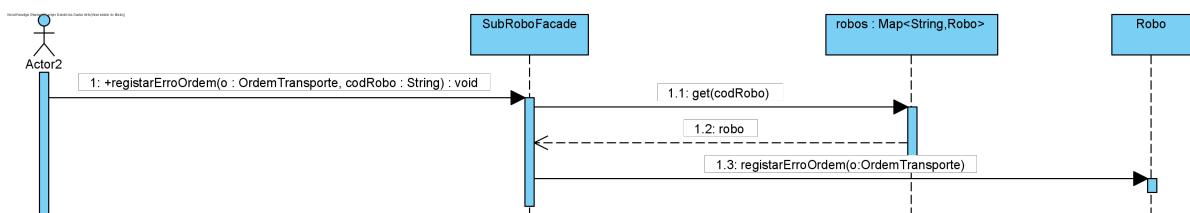
### 5.2.16 registrarErroPalete



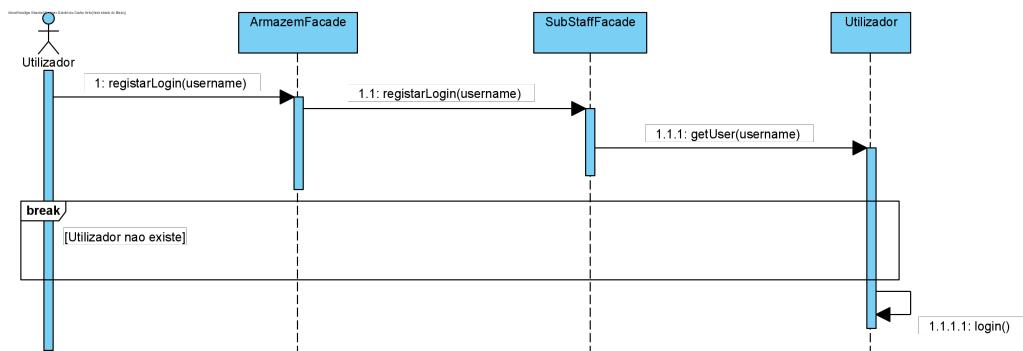
### 5.2.17 registrarConsulta



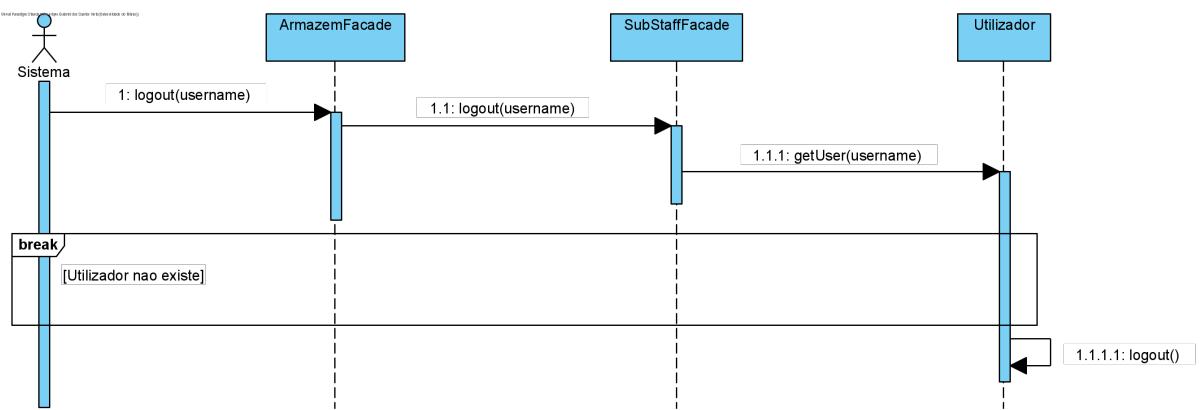
### 5.2.18 registrarErroOrdem



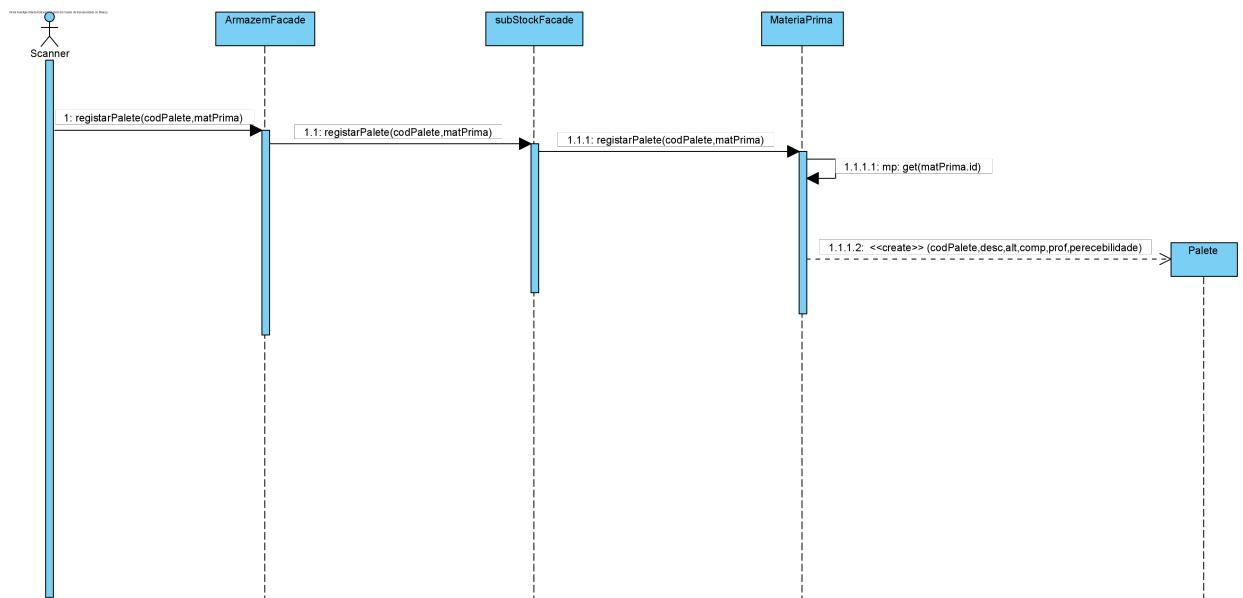
### 5.2.19 registrarLogin



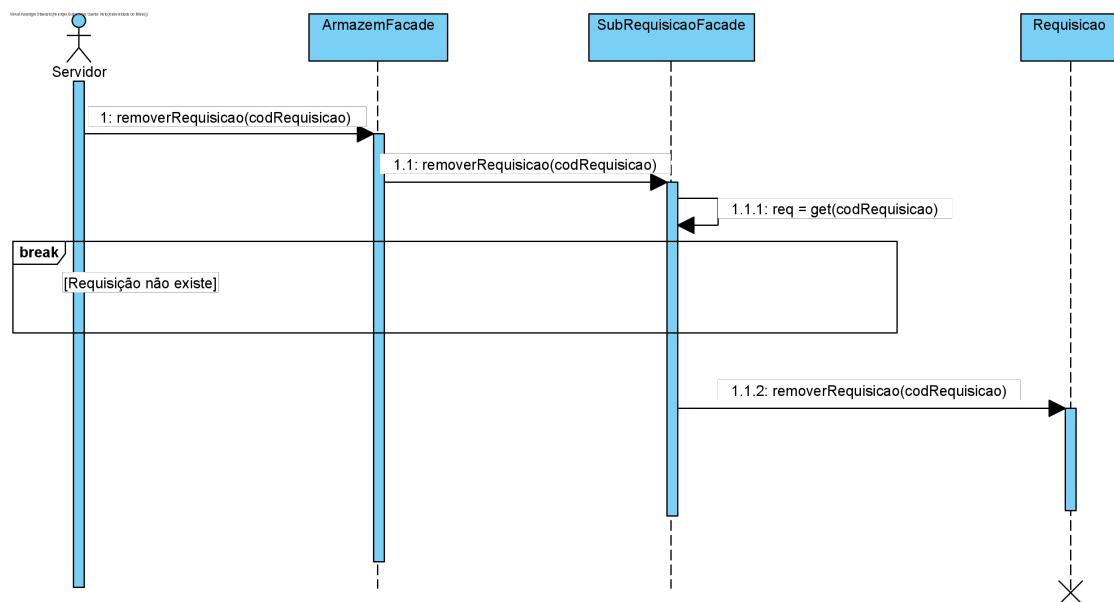
### 5.2.20 registrarLogout



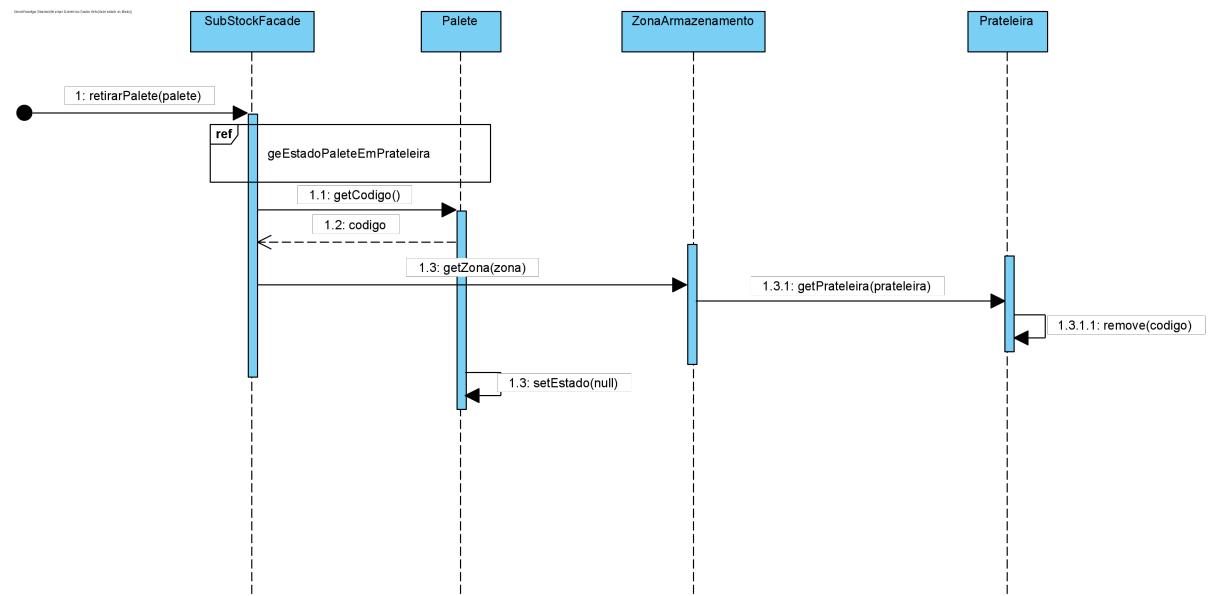
### 5.2.21 registrarPalete



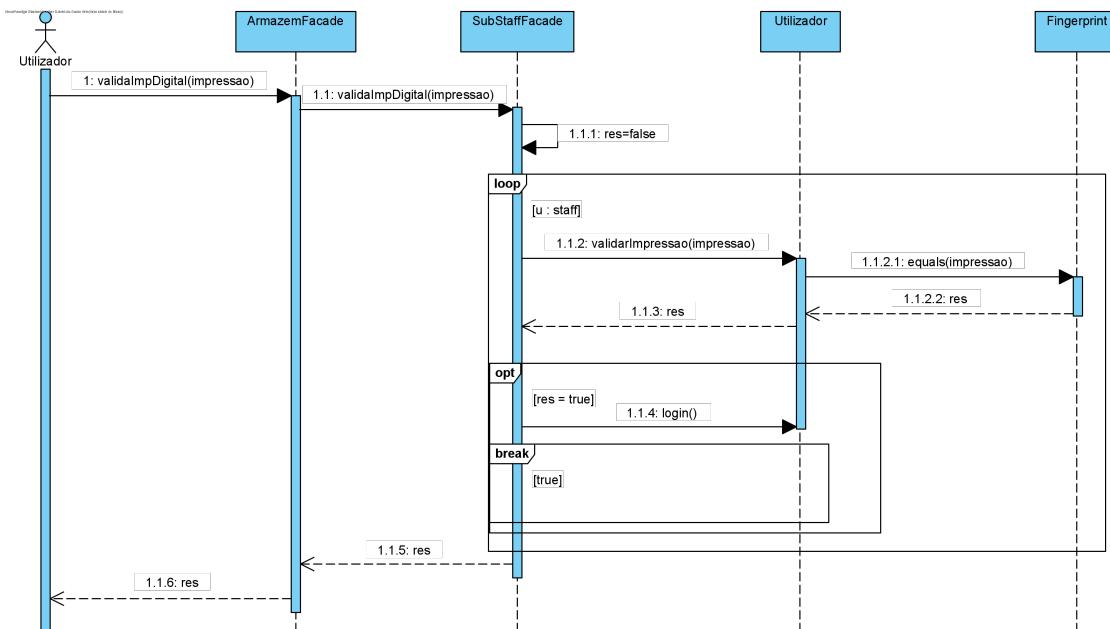
### 5.2.22 removerRequisicao



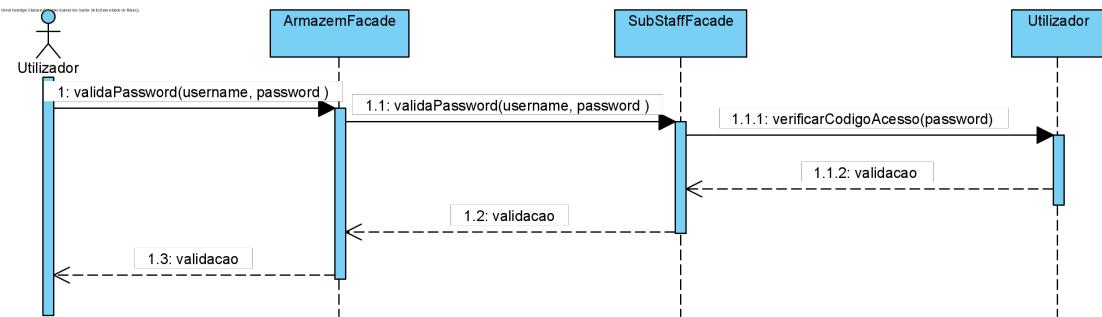
### 5.2.23 retirarPalete



### 5.2.24 validaImpDigital



### 5.2.25 ValidaPassword



### 5.2.26 validaUsername

