

UNIVERSIDADE DO MINHO

DEPARTAMENTO DE INFORMÁTICA

Fase 2  
Grupo N<sup>o</sup> 4

João Correia (A84414)

Marco Pereira (A89556)

Pedro António (A58062)

Rúben Cerqueira (A89593)

28 de novembro de 2020



João



Marco



Pedro



Rúben

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>4</b>
<b>2</b>	<b>Alterações à primeira fase</b>	<b>5</b>
2.1	Modelo de Domínio . . . . .	5
2.2	Use Cases . . . . .	6
<b>3</b>	<b>Diagrama de Componentes</b>	<b>7</b>
<b>4</b>	<b>Diagrama de Classes</b>	<b>9</b>
<b>5</b>	<b>Diagramas de Sequência</b>	<b>11</b>
<b>6</b>	<b>Conclusão</b>	<b>12</b>
<b>A</b>	<b>Diagrama de Componentes</b>	<b>13</b>
<b>B</b>	<b>Diagrama de Classes</b>	<b>14</b>
<b>C</b>	<b>Diagramas de Sequência</b>	<b>15</b>

# **Lista de Figuras**

2.1	Modelo de Domínio alterado . . . . .	5
3.1	Diagrama de Componentes . . . . .	8
4.1	Diagrama de Classes . . . . .	10
A.1	Diagrama de Componentes . . . . .	13
B.1	Diagrama de Classes . . . . .	14
C.1	Diagrama de Sequência do método adicionaCodigoQREncarregado . . . . .	15
C.2	Diagrama de Sequência do método adicionaPedidoRequisicao . . . . .	15
C.3	Diagrama de Sequência do método calculaRotaRobo . . . . .	16
C.4	Diagrama de Sequência do método comunicaCodigoQR . . . . .	16
C.5	Diagrama de Sequência do método comunicaInformacaoTransporte . . . . .	17
C.6	Diagrama de Sequência do método defineUtilizadorAtivo . . . . .	17
C.7	Diagrama de Sequência do método depthFirst . . . . .	18
C.8	Diagrama de Sequência do método encontraPrateleira . . . . .	18
C.9	Diagrama de Sequência do método encontraRoboLivre . . . . .	19
C.10	Diagrama de Sequência do método getPaletes . . . . .	19
C.11	Diagrama de Sequência do método getPaletesRobo . . . . .	20
C.12	Diagrama de Sequência do método getPaletesStock . . . . .	20
C.13	Diagrama de Sequência do método getPartePedidoDisponivel . . . . .	21
C.14	Diagrama de Sequência do método getPartePedidoNaoDisponivel . . . . .	21
C.15	Diagrama de Sequência do método login . . . . .	22
C.16	Diagrama de Sequência do método logout . . . . .	22
C.17	Diagrama de Sequência do método passwordCorrecta . . . . .	23
C.18	Diagrama de Sequência do método verificaPedidoDisponivel . . . . .	23

# Capítulo 1

## Introdução

O presente relatório visa apresentar a segunda fase do trabalho proposto no âmbito da Unidade Curricular de Desenvolvimento de Sistemas de Software.

A segunda fase do projecto tem continuidade direta da primeira fase, em que foram concretizados os **Modelos de Domínio** e os **Modelos de Use Cases**, e vai ser a partir destes que vamos continuar a modelar a gestão do nosso Armazém Inteligente. Nesta fase foram feitas modelações mais concretas deste: o **Diagrama de Classes**, o **Diagrama de Componentes** e os vários **Diagramas de Sequência**.

O **Diagrama de Componentes** é uma modelação que consiste na divisão do projeto em vários sistemas que têm um funcionamento interno autónomo e capazes de interagir com os outros componentes por via a realizar o modelo proposto.

É neste ponto que podemos começar a pensar com muito mais detalhe na implementação que queremos executar, incluindo a escolha da linguagem de programação na qual esta vai ser elaborada. O próximo passo é intuitivamente o **Diagrama de Classes**, tendo agora em vista que a implementação do armazém será feita numa linguagem orientada aos objetos, e visa detalhar o funcionamento interno de cada um dos subsistemas propostos no **Diagrama de Componentes**.

Para finalizar esta fase do projeto, procedeu-se à elaboração dos **Diagramas de Sequência**, que já apresentam o funcionamento de todo o processo ao nível da linguagem de programação, ainda que de forma esquemática. Daremos uma explicação mais detalhada da finalidade e estrutura de cada um dos diagramas apresentados nas suas respetivas secções do relatório.

Ao longo da elaboração dos diagramas foram feitas várias mudanças nos **Modelos de Domínio** e nos **Modelos de Use Cases** para que conseguíssemos obter a solução mais correta possível, e portanto, as suas alterações tal como os motivos que nos levaram a fazê-las serão abordadas posteriormente.

Começaremos assim por fazer uma breve abordagem às alterações efetuadas aos diagramas entregues na primeira fase do projeto.

# Capítulo 2

## Alterações à primeira fase

Conforme fomos avançando nesta segunda fase do projeto, foi necessário fazer alterações aos diagramas previamente propostos. À medida que vamos interagindo com os processos a um detalhe mais fino, vamo-nos apercebendo que existem elementos que não estavam ainda presentes, ou cuja função vai ser modificada.

Todas as alterações efetuadas estão assim dispostas abaixo:

### 2.1 Modelo de Domínio

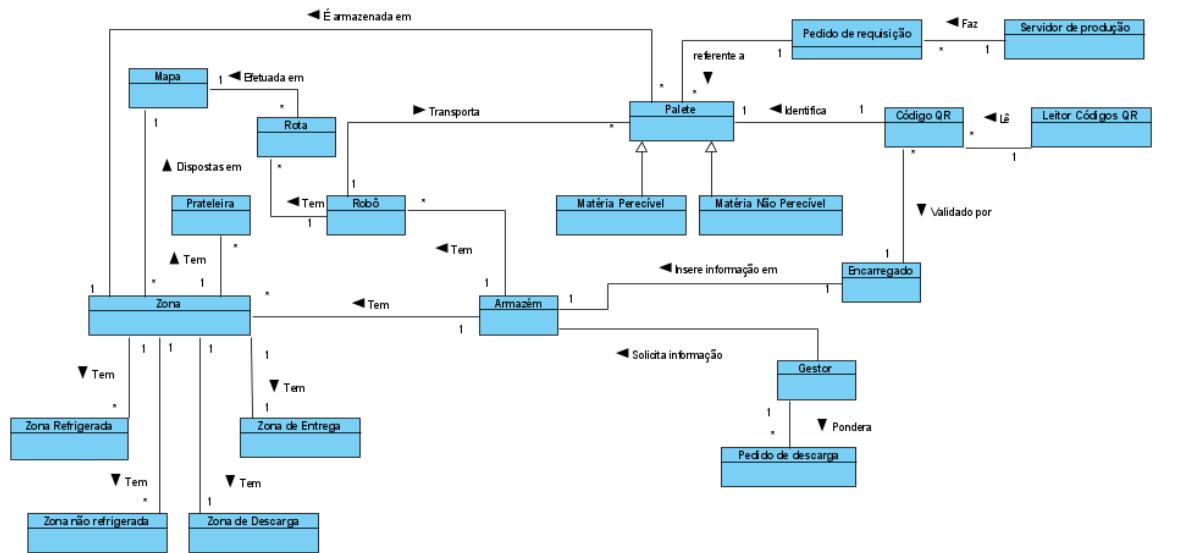


Figura 2.1: Modelo de Domínio alterado

Em relação ao **Modelo de Domínio** entregue na primeira fase, foram feitas as seguintes alterações:

- **Encarregado** - Foi introduzido um novo tipo de utilizador, para validar os códigos QR após a sua leitura, e subsequentemente inserir esta informação no sistema.

- **Zona** - As Zonas Refrigeradas e Zonas Não Refrigeradas passam agora a fazer parte de uma entidade mais genérica, assim como as Zonas de Descarga e Entrega. Esta mudança foi necessária pois todas estas Zonas têm agora Prateleiras.
- **Mapa** - Foi necessário implementar uma entidade para conter o mapa do nosso armazém, definido como um grafo.
- **Rota** - Para que os Robôs saibam para onde se têm que dirigir no grafo que representa o nosso Mapa, adicionou-se uma entidade Rota, que vai conter as direções que o Robô terá que tomar para chegar ao seu destino.
- **Motorista** - A entidade Motorista foi retirada do Modelo de Domínio pois ao longo da segunda fase este revelou-se externo a todo o processo de organização do Armazém.

## 2.2 Use Cases

Houve ainda algumas alterações aos **Use Cases**, uma vez que os fluxos de funcionamento do Armazém ganharam agora mais complexidade, a qual só nos foi aparente após a elaboração tanto do **Diagrama de Classe** como dos **Diagramas de Sequência**.

Seguidamente vamos analisar as correções efetuadas:

- **Solicitar Autorização de Descarga (Motorista)** - Uma vez que a entidade Motorista já não faz parte do nosso Sistema, o Use Case em que este era o ator é eliminado.
- **Transportar Paleta (Robô)** - Após o design dos Diagramas de Sequência foi uma decisão intuitiva eliminar este Use Case e adicionar as suas responsabilidades ao Use Case **Notificar Conclusão de Transporte**, uma vez que o Use Case anterior ficava agora sem respostas por parte do Sistema e portanto desprezável.

Passamos seguidamente a dar uma abordagem mais pormenorizada de cada um dos diagramas propostos para a segunda fase.

## Capítulo 3

# Diagrama de Componentes

O **Diagrama de Componentes** foi o passo intuitivo após a primeira fase do Projeto. Este tem por função organizar o modelo em subsistemas com algum grau de complexidade, em que cada um deles desempenha uma função distinta. Esta vista ainda de alto nível permite-nos ter uma ideia do funcionamento geral da aplicação sem entrar em demasiado pormenor, o que é extremamente útil numa fase em que é necessário decidir como vai ser a organização geral do sistema. Após a sua formulação estaríamos aptos para começar a mostrar ao cliente qual será a solução adoptada para a resolução do problema, de uma forma simples mas que sintetiza a totalidade do projeto a desenvolver.

Após uma análise detalhada dos modelos anteriores surgiram os seguintes subsistemas:

- **Robo** - Vai tratar de toda a gestão dos Robôs, saber quais estão disponíveis, as suas localizações, lidar com as notificações que estes enviam, assim como calcular as Rotas para que estes possam percorrer o Armazém de forma eficiente e sem colisões com outros Robôs.
- **Stock** - Este Subsistema vai gerir o armazenamento das Paletes em Prateleiras, divididas pelas diferentes Zonas, podendo dar indicações quanto à disponibilidade de espaço ou de matérias primas no Armazém.
- **Utilizador** - Gere a ligação de todos os tipos de utilizadores à aplicação, permitindo efetuar o login, logout, verificação de password, assim como definir a que funções da aplicação cada utilizador tem acesso.
- **Pedido** - Vai conter toda a lógica relacionada com os Pedidos de Descarga bem como Pedidos de Entrega, até que estes sejam concretizados ou rejeitados.

Esta configuração de subsistemas será assim a base para a nossa organização, e pode ser visualizada no **Diagrama de Componentes** abaixo:

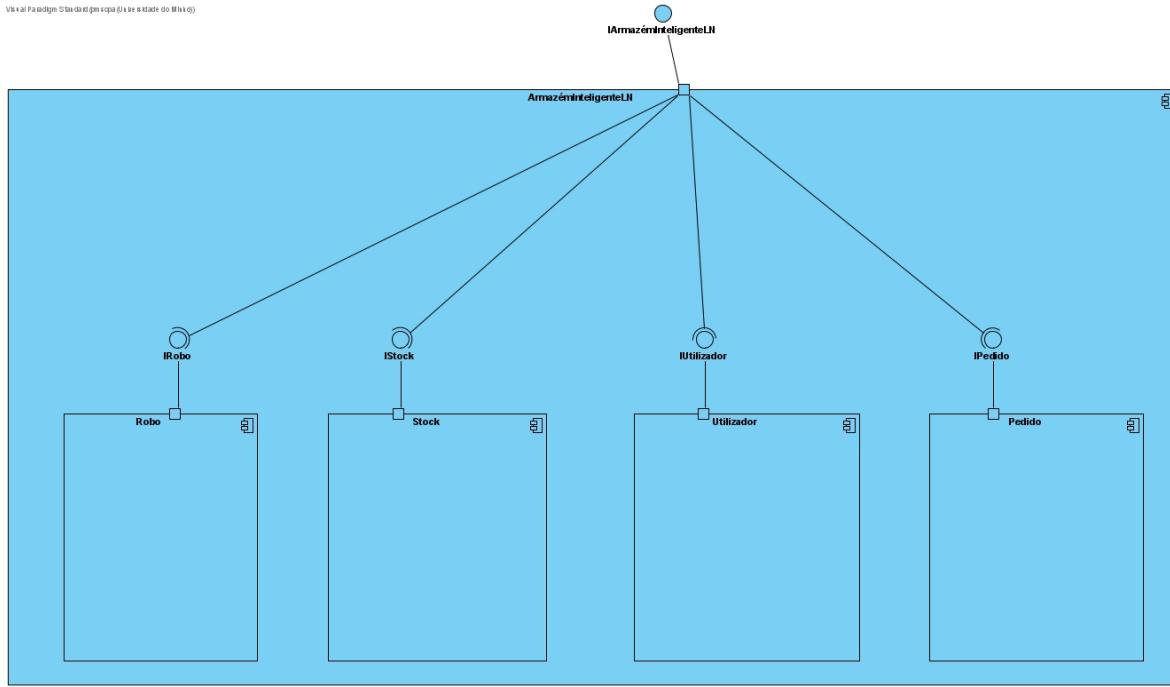


Figura 3.1: Diagrama de Componentes

Para garantir que cada Subsistema funciona de forma completamente encapsulada, são implementadas **Facades**, que vão fazer a mediação entre a organização interna, e a interação com outros Subsistemas.

Deste modo, os outros Subsistemas apenas têm que saber a API da Facade para poder interagir, mesmo sem conhecer a organização interna deste Subsistema.

## Capítulo 4

# Diagrama de Classes

O **Diagrama de Classes** é algo com o qual já estamos acostumados, uma vez que foi usado de forma extensa no decorrer das aulas de Programação Orientada aos Objetos. Após a definição do **Diagrama de Componentes** podemos já decidir que classes vamos usar dentro de cada um dos subsistemas.

Aqui já sabemos que vamos utilizar uma linguagem de programação orientada aos objetos, e podemos assim já detalhar todas as interfaces que vamos usar, bem como métodos e variáveis de instância de cada uma das classes que decidimos incorporar na nossa implementação do Armazém Inteligente.

O Diagrama está organizado por Subsistemas, cada um destes inserido no seu Package. Cada Package contém todas as classes que são necessárias para o seu funcionamento interno, e ainda uma Facade, que vai implementar uma interface (API) que será conhecida pelos outros Subsistemas, e com a qual eles poderão comunicar sem acesso a qualquer uma das estruturas contidas dentro de cada Package.

Esta implementação garante que não só temos a certeza que só disponibilizamos a informação estritamente necessária ao exterior de cada Subsistema, mas que qualquer Subsistema que implemente a mesma API poderia ser usado em lugar dos Subsistemas presentes, sem que isso afetasse o funcionamento pleno do Armazém.

Este diagrama já começa a descrever o fluxo que o nosso programa vai ter, e dado que estamos a um nível mais baixo que as representações anteriores, há agora a necessidade acrescida de garantir que todas as formulações presentes nos diagramas anteriores são cumpridas.

O Diagrama de Classes que foi implementado pelo grupo nesta fase é o seguinte:

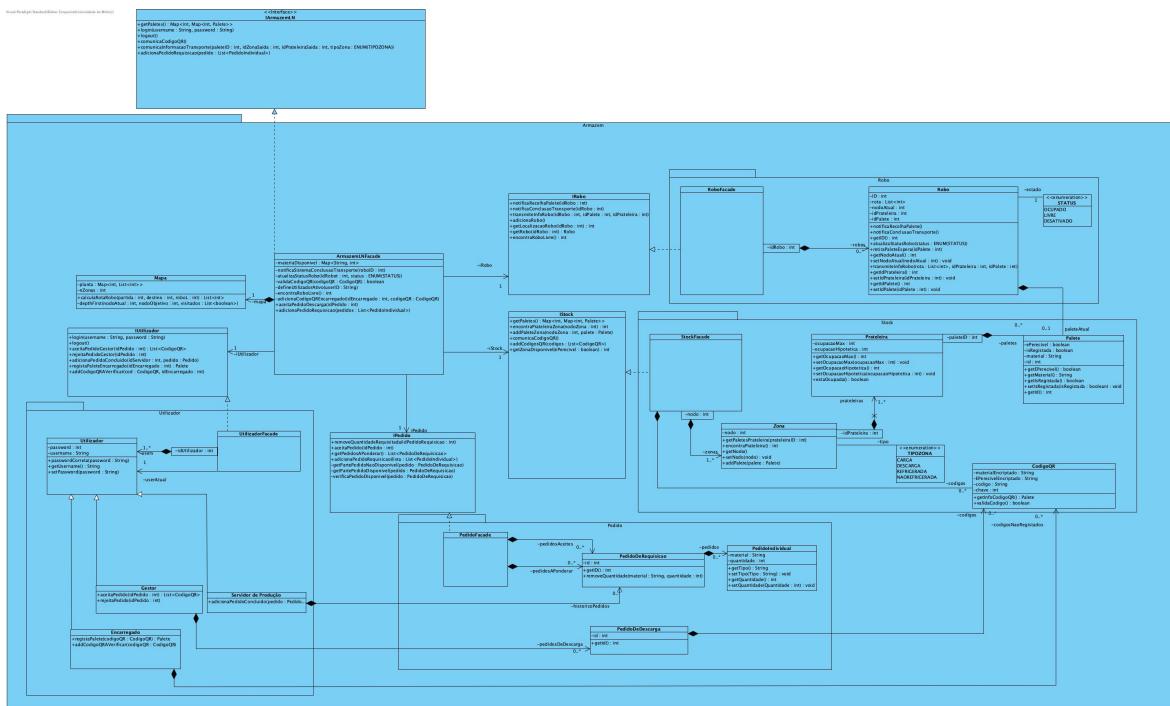


Figura 4.1: Diagrama de Classes

## Capítulo 5

# Diagramas de Sequência

O último tipo de diagrama a desenvolver nesta fase do projeto é o **Diagrama de Sequência**.

Dada a complexidade do projeto, seria inviável tentar descrever todo o seu funcionamento apenas num diagrama. Surge assim a necessidade da utilização de Diagramas de Sequência. Estes descrevem o fluxo das interações entre os nossos objetos de uma forma simples, e já implementam o conceito de ordem pela qual estas têm lugar. É crucial ter uma ideia bem definida da sequência pela qual o nosso projeto vai realizar as suas operações, antes de começar realmente a programa-lo.

Os Diagramas de Sequência já estão a um nível que pode ser facilmente convertido para código, mas com uma apresentação mais intuitiva. Tal facto levou a que identificássemos problemas nas formulações anteriores, e confere-nos a certeza de que o programa está a seguir os passos pretendidos, produzindo assim os resultados esperados.

No Apêndice C estão dispostos todos os Diagramas de Sequência produzidos pelo grupo.

# Capítulo 6

## Conclusão

Após esta segunda fase do projeto estamos em posição de realmente começar a escrever o código para a aplicação pretendida. Tal facto deve-se ao desenvolvimento faseado que foi implementado, e que nos permite garantir que a base para o início de cada subsequente iteração corresponde aos requisitos exigidos.

Tendo o trabalho desenvolvido na primeira fase como base, dividiu-se o projeto em subsistemas para tratar das diversas facetas presentes na organização do armazém. A partir desta divisão foi possível ponderar que classes iriam ser utilizadas, e chegou-se assim a um Diagrama de Classes já com as variáveis e métodos que cada objeto vai implementar. Por fim, temos já todos os Diagramas de Sequência para detalhar o funcionamento do nosso programa de forma integral.

Assim, a parte da implementação do projeto em código Java vai ser bastante suave, uma vez que a escrita do seu código a partir dos diagramas vai ser praticamente imediata. Tal remete-nos para a mensagem que com um planeamento sólido, a parte de programar a aplicação é na verdade a mais simples, pois já temos todos os dados necessários para que não haja reformulações na estrutura do projeto. Esta certeza já nos dá a confiança de que todos os elementos vão conseguir ter as interações desejadas sem termos que voltar atrás durante a escrita do código.

Constatou-se que por diversas vezes houve necessidade de alterar Diagramas anteriores porque a nossa abordagem se tornou inexequível, o que se revelou num processo relativamente simples. Percebemos assim que se essa alteração tivesse que ser feita já após a escrita do código, seria um processo bastante mais laborioso e com uma probabilidade de erro extremamente alta.

Deste modo, avançamos para a terceira fase com uma ideia concreta da aplicação que queremos programar. Esta já vai consistir apenas na implementação da visão que já desenhámos, e não em algo que ainda seja propício a alterações.

Há um velho ditado que uma imagem vale mil palavras, e o que esta segunda fase do projeto nos ensinou é que um bom diagrama vale também mil linhas de código.

## Apêndice A

# Diagrama de Componentes

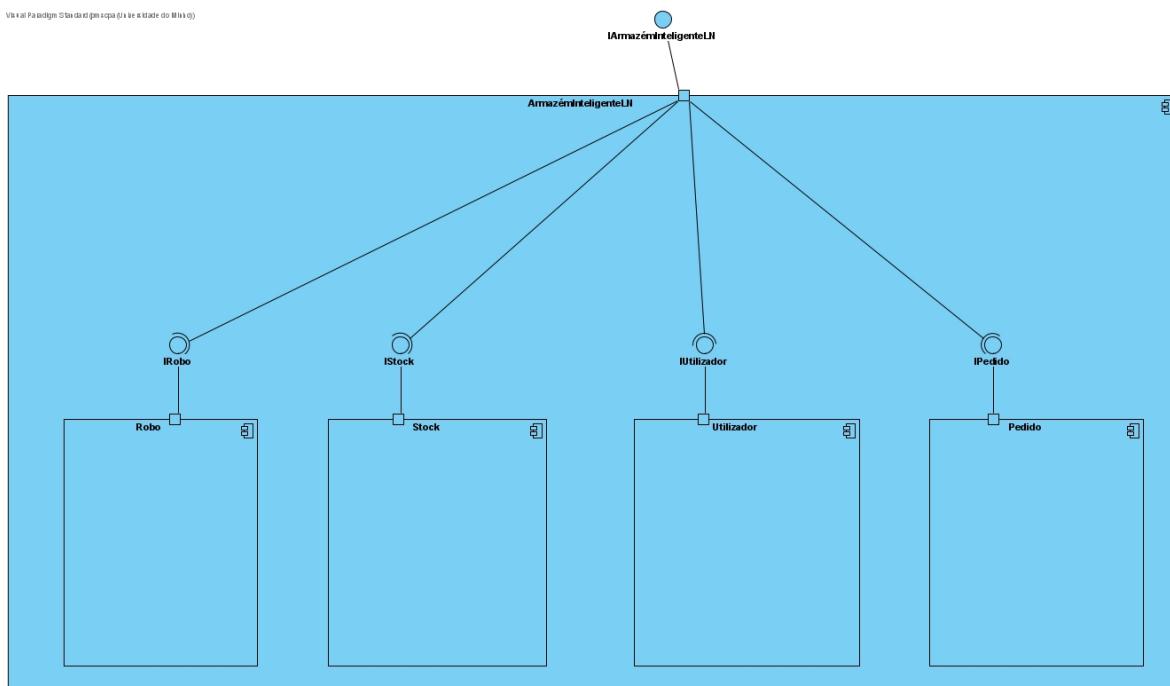


Figura A.1: Diagrama de Componentes

# Apêndice B

# Diagrama de Classes

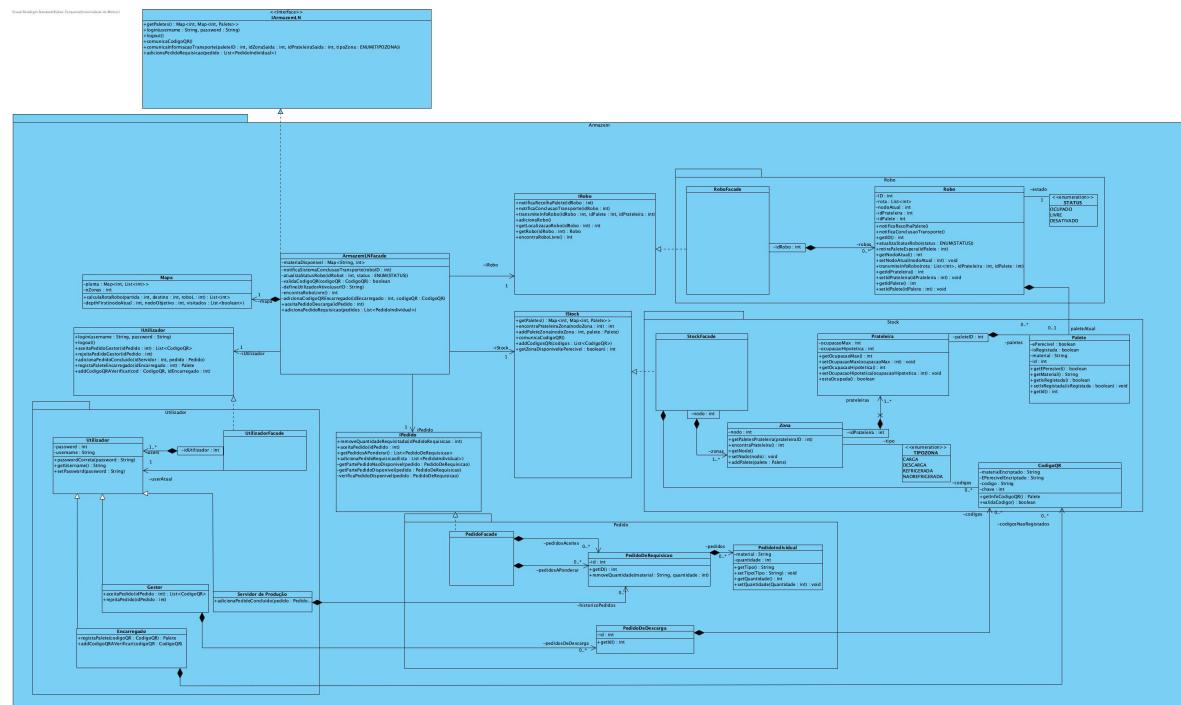


Figura B.1: Diagrama de Classes

## Apêndice C

# Diagramas de Sequência

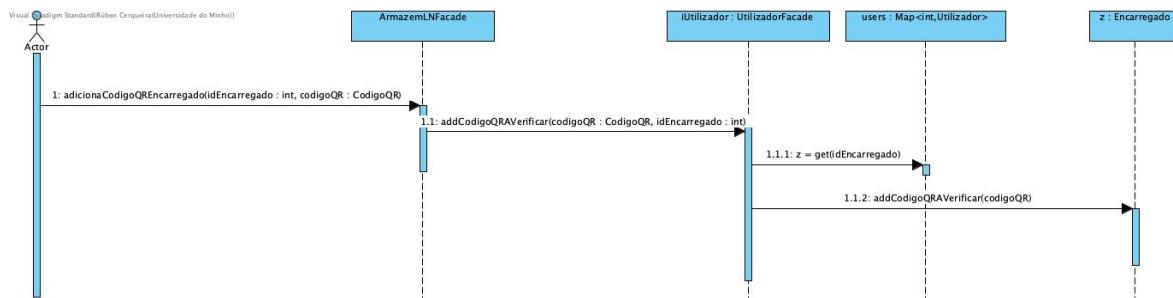


Figura C.1: Diagrama de Sequência do método `adicionaCodigoQREncarregado`

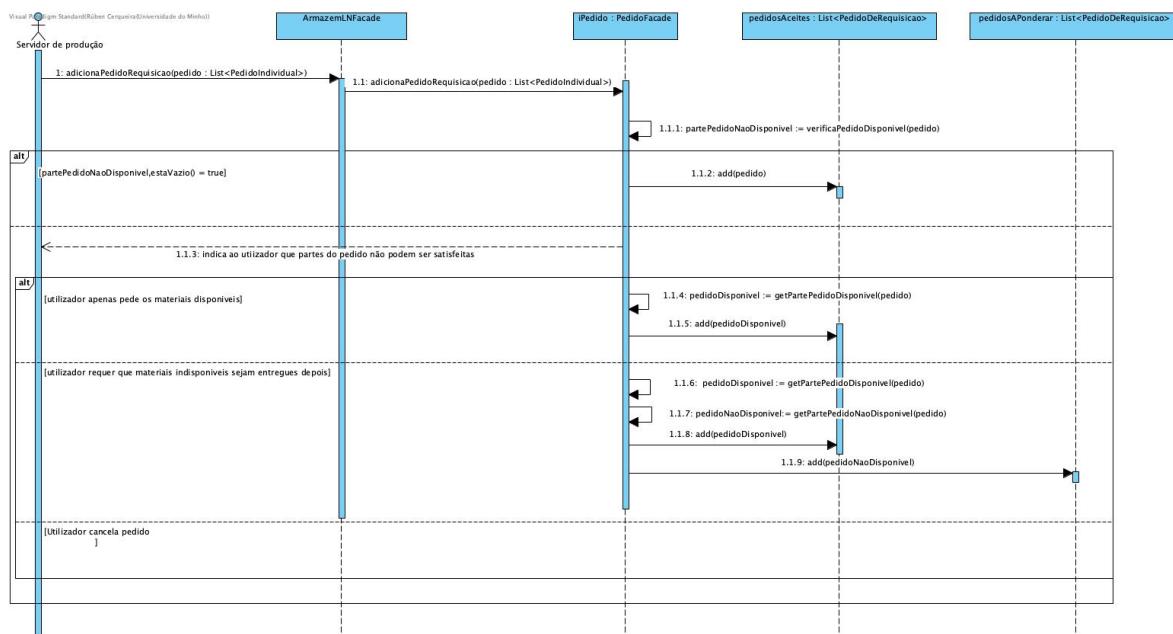


Figura C.2: Diagrama de Sequência do método `adicionaPedidoRequisicao`

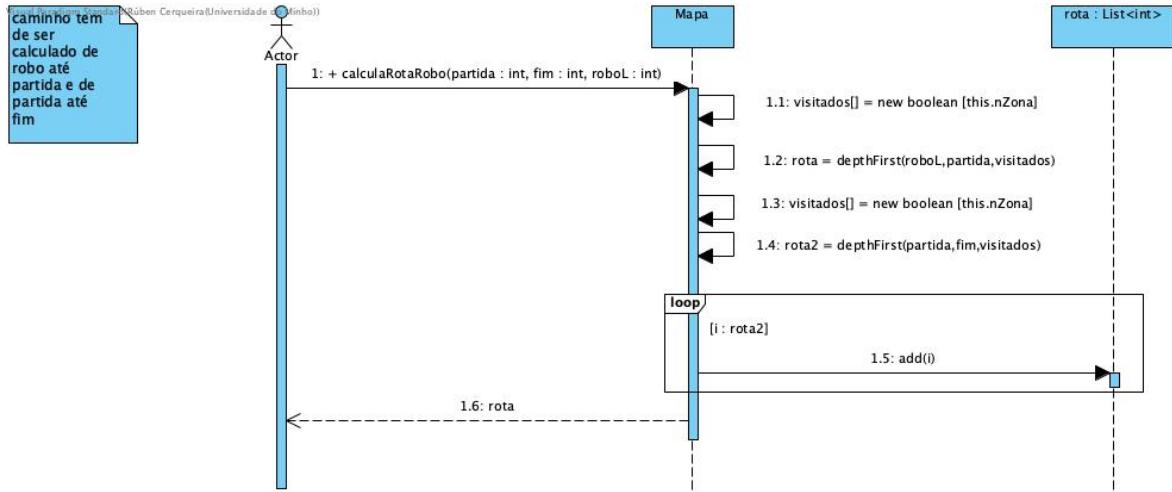


Figura C.3: Diagrama de Sequência do método calculaRotaRobo

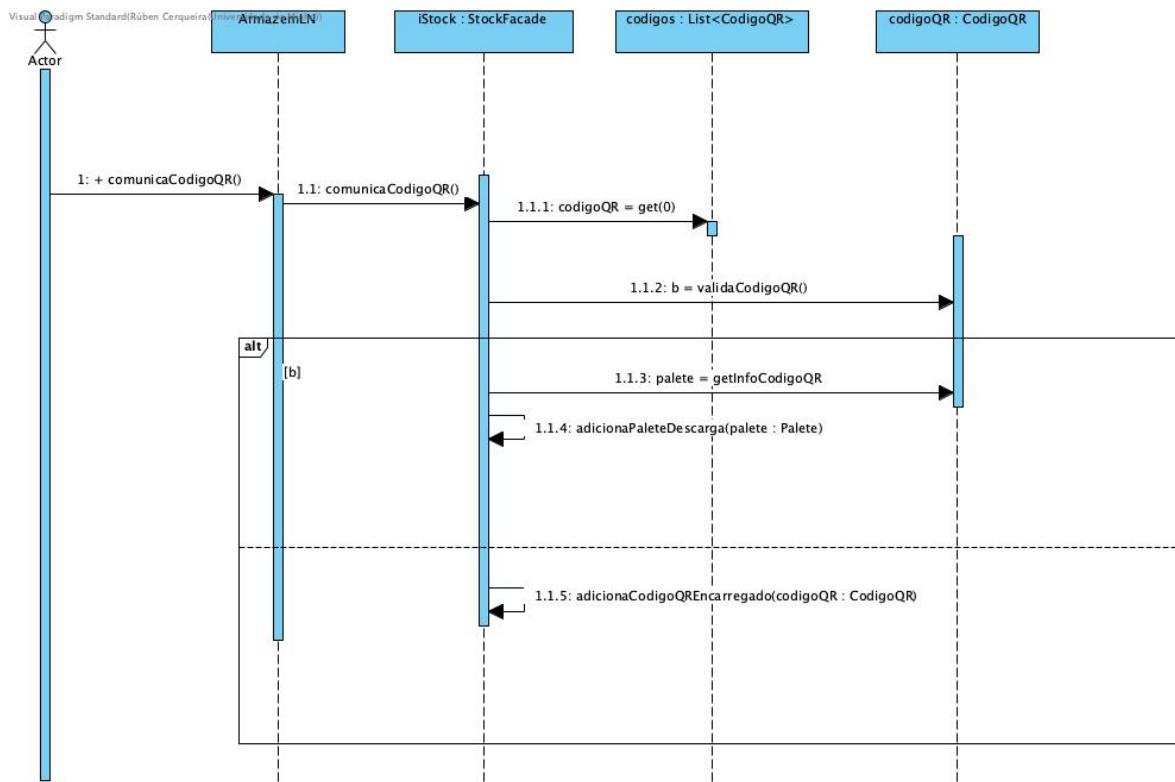


Figura C.4: Diagrama de Sequência do método comunicaCodigoQR

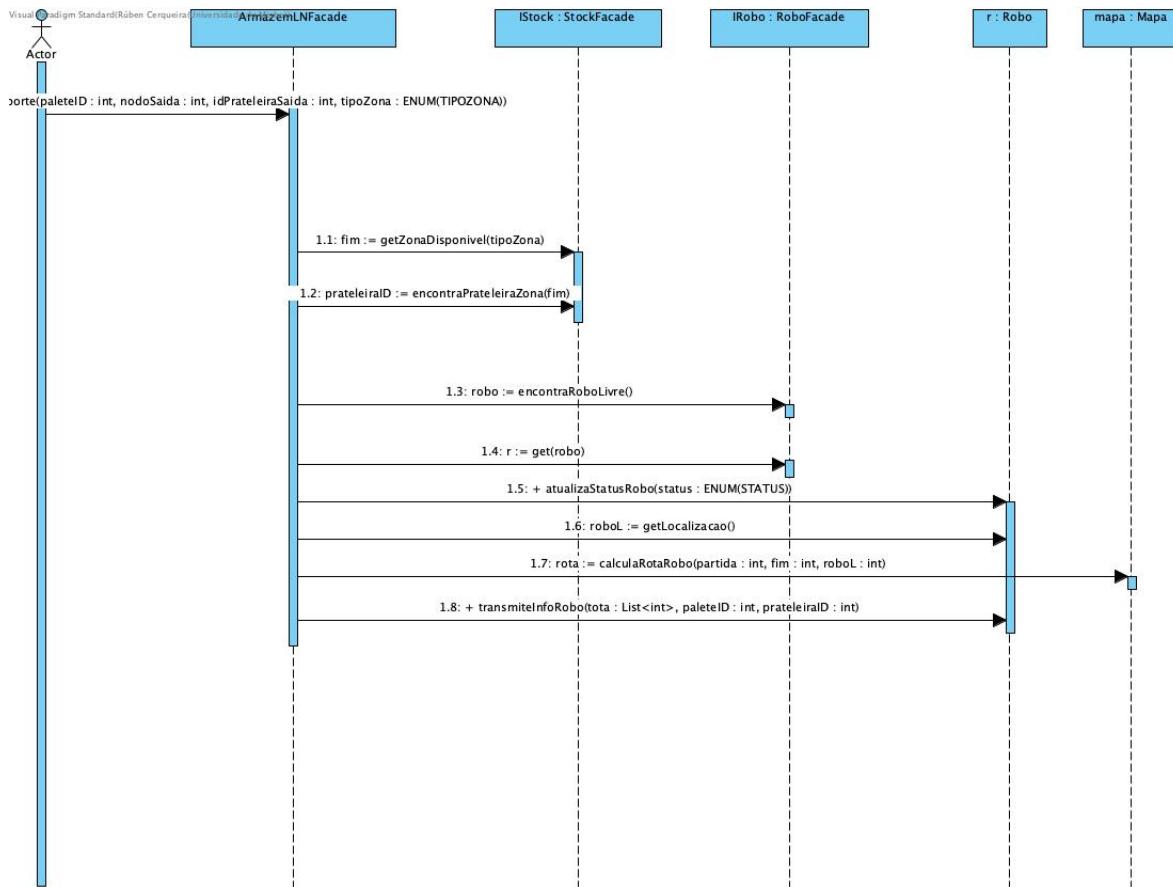


Figura C.5: Diagrama de Sequência do método comunicaInformacaoTransporte

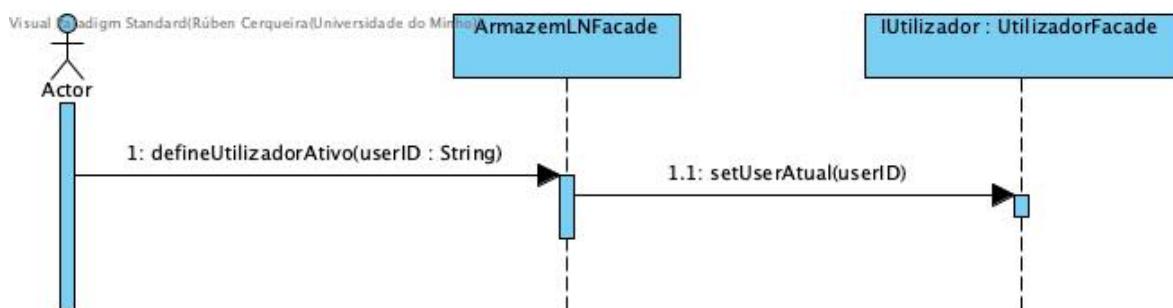


Figura C.6: Diagrama de Sequência do método defineUtilizadorAtivo

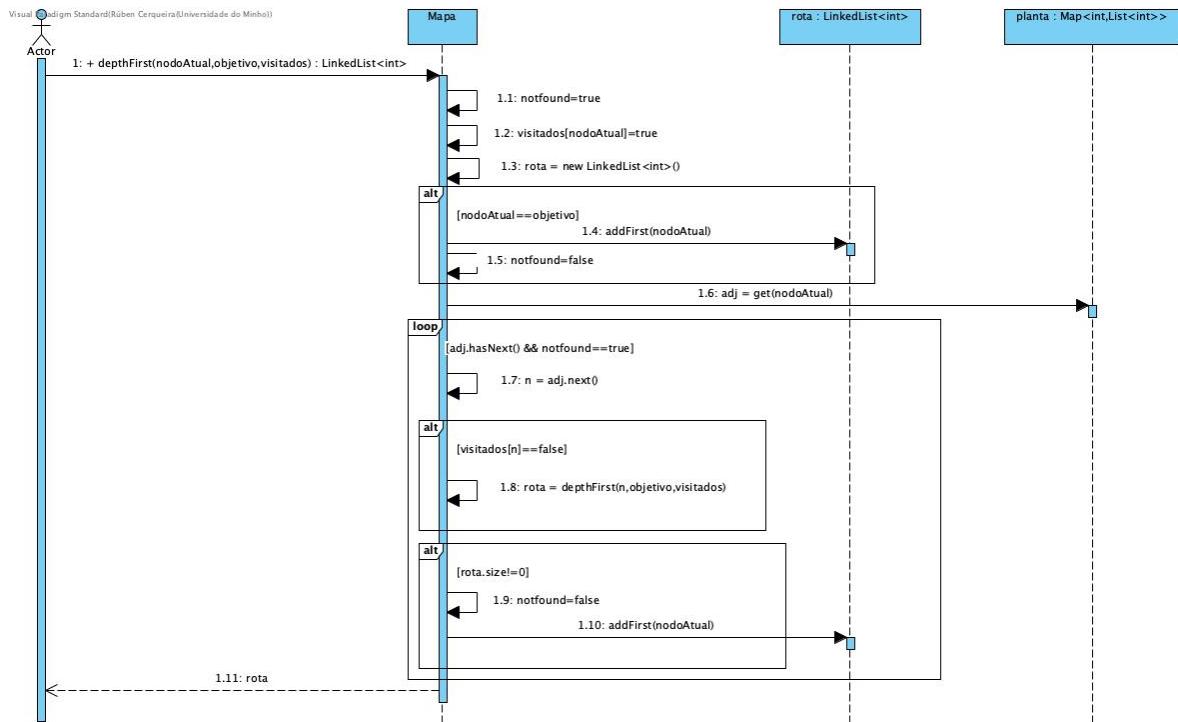


Figura C.7: Diagrama de Sequência do método `depthFirst`

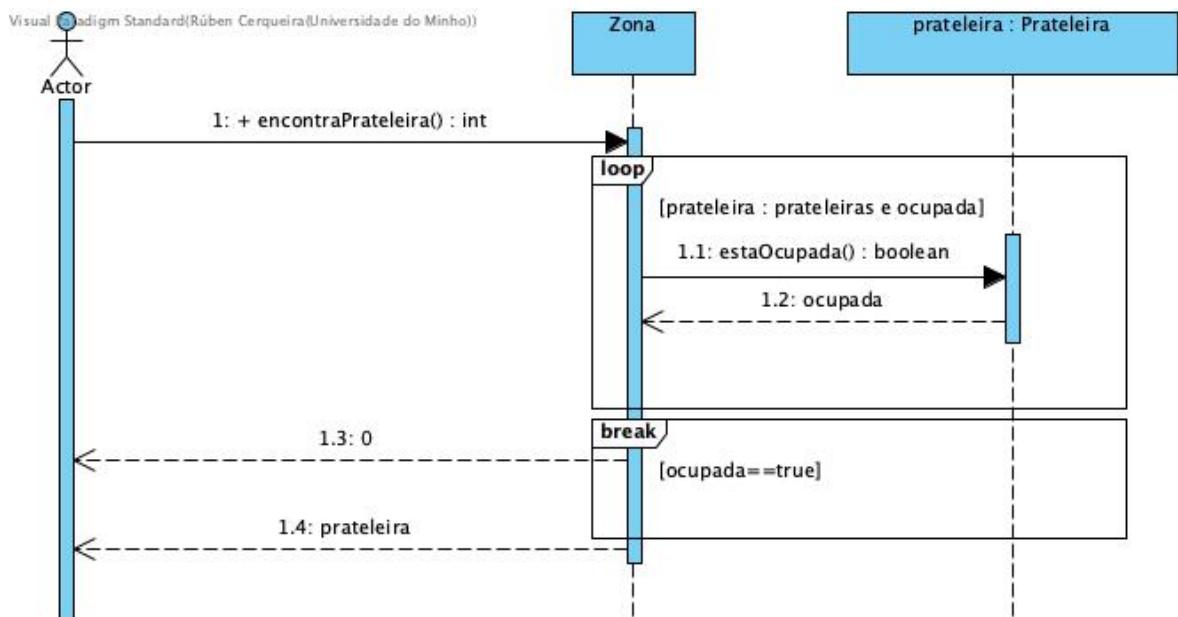


Figura C.8: Diagrama de Sequência do método `encontraPrateleira`

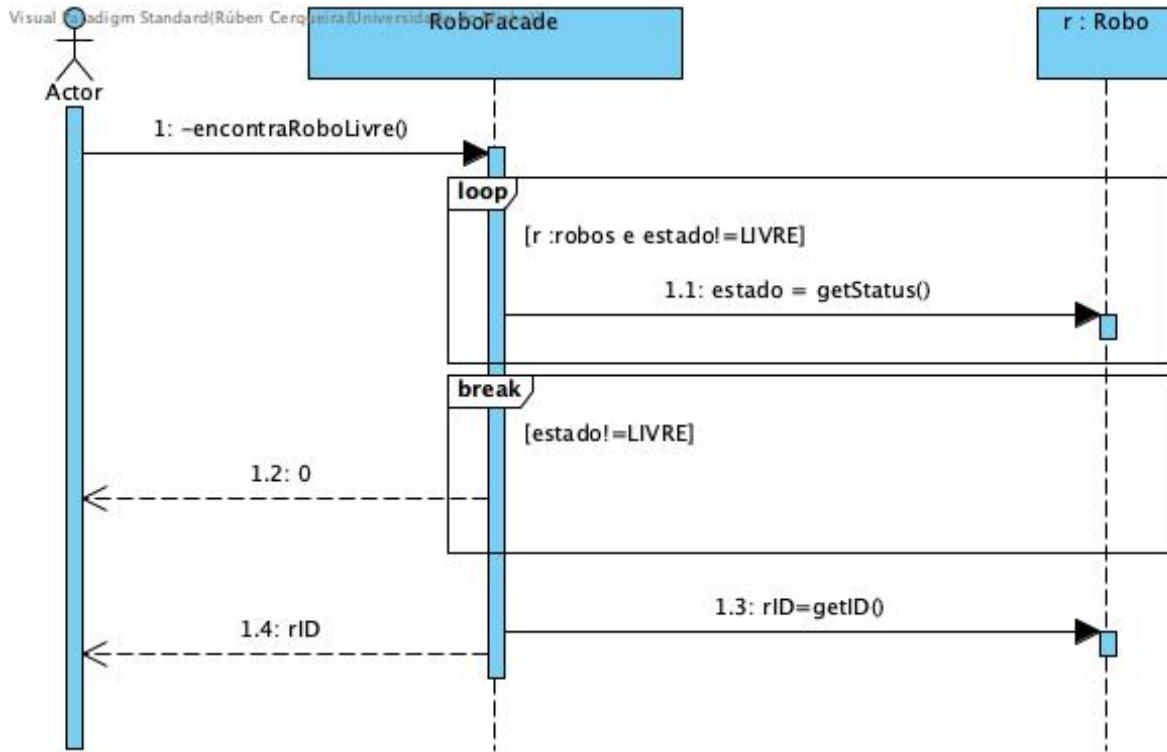


Figura C.9: Diagrama de Sequênciа do m todo encontraroBoboLivre

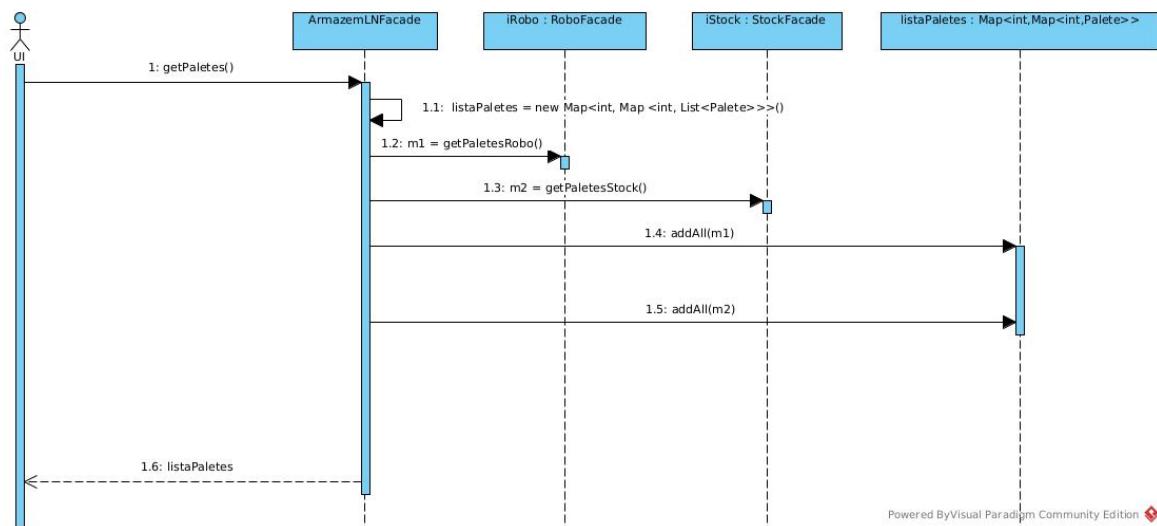


Figura C.10: Diagrama de Sequênciа do m todo getPaletes

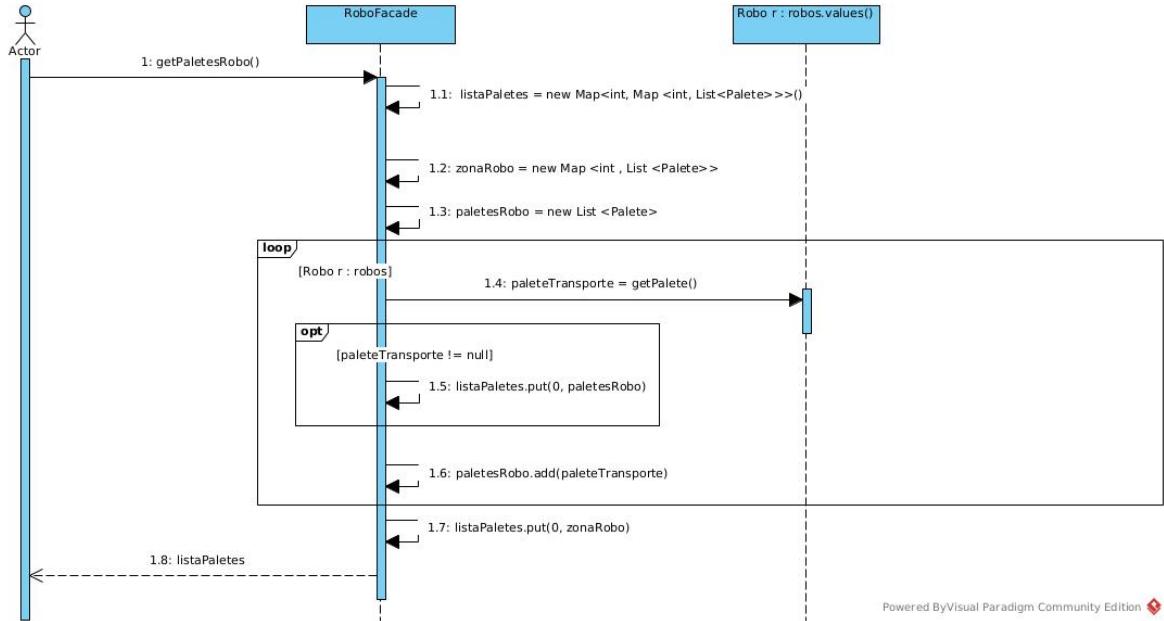


Figura C.11: Diagrama de Sequência do método getPaletesRobo

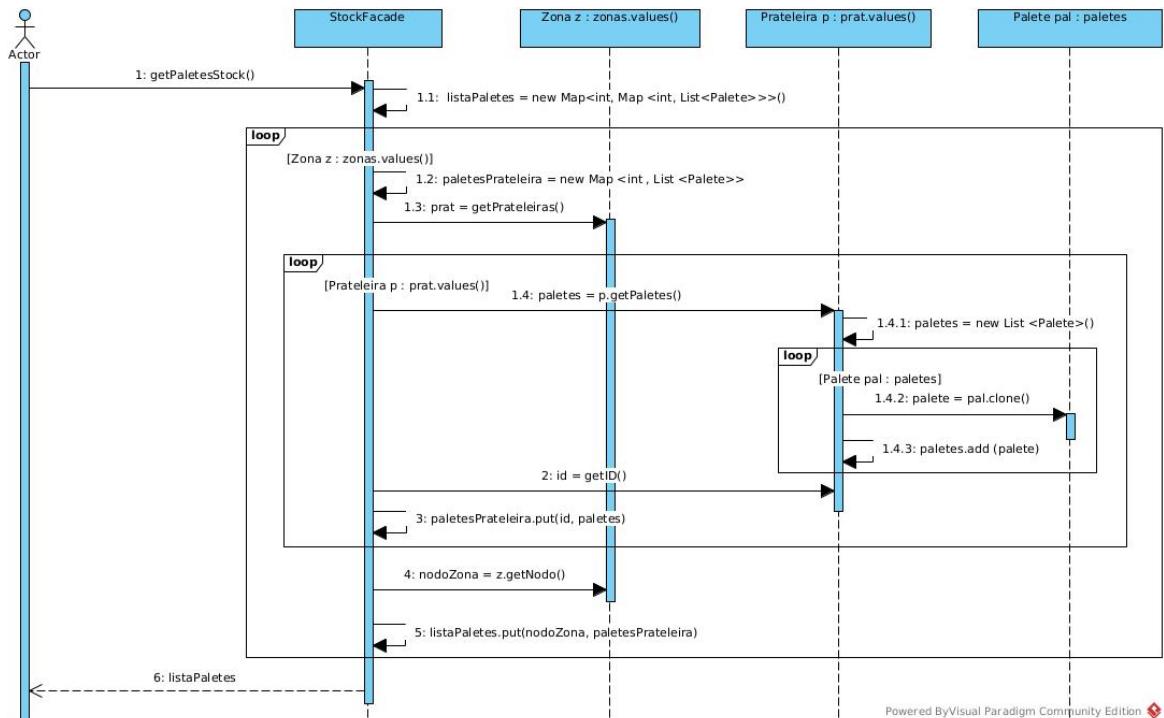


Figura C.12: Diagrama de Sequência do método getPaletesStock

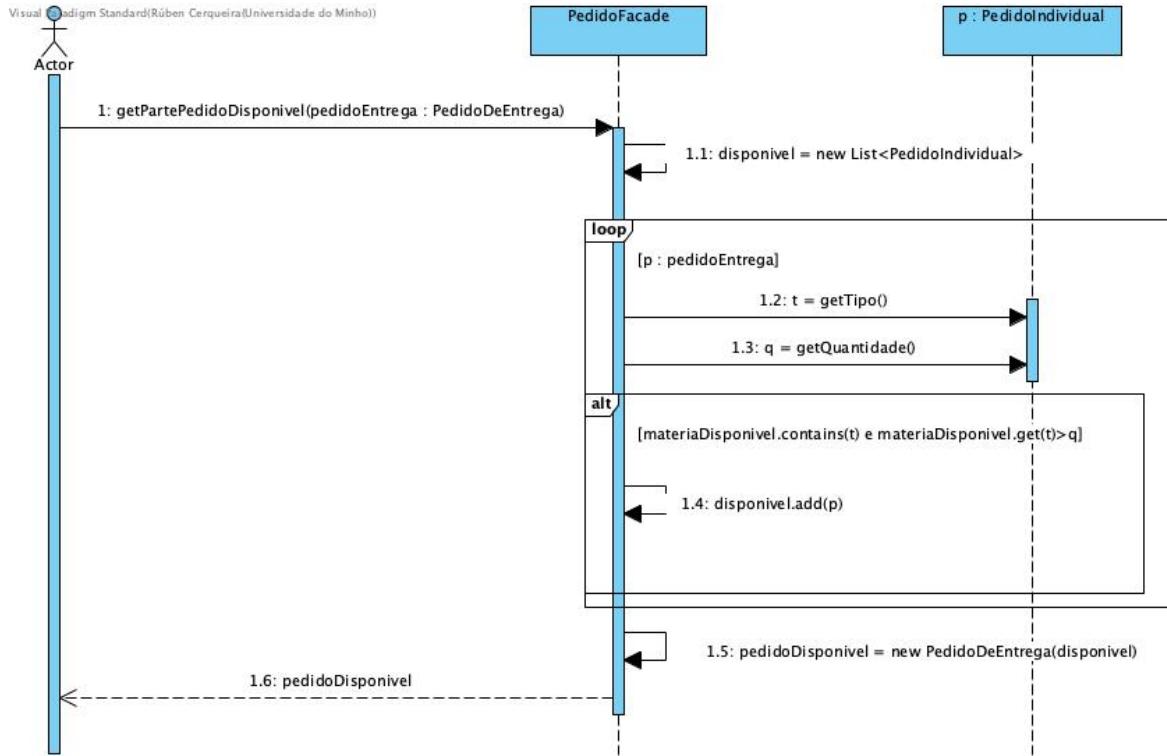


Figura C.13: Diagrama de Sequência do método `getPartePedidoDisponivel`

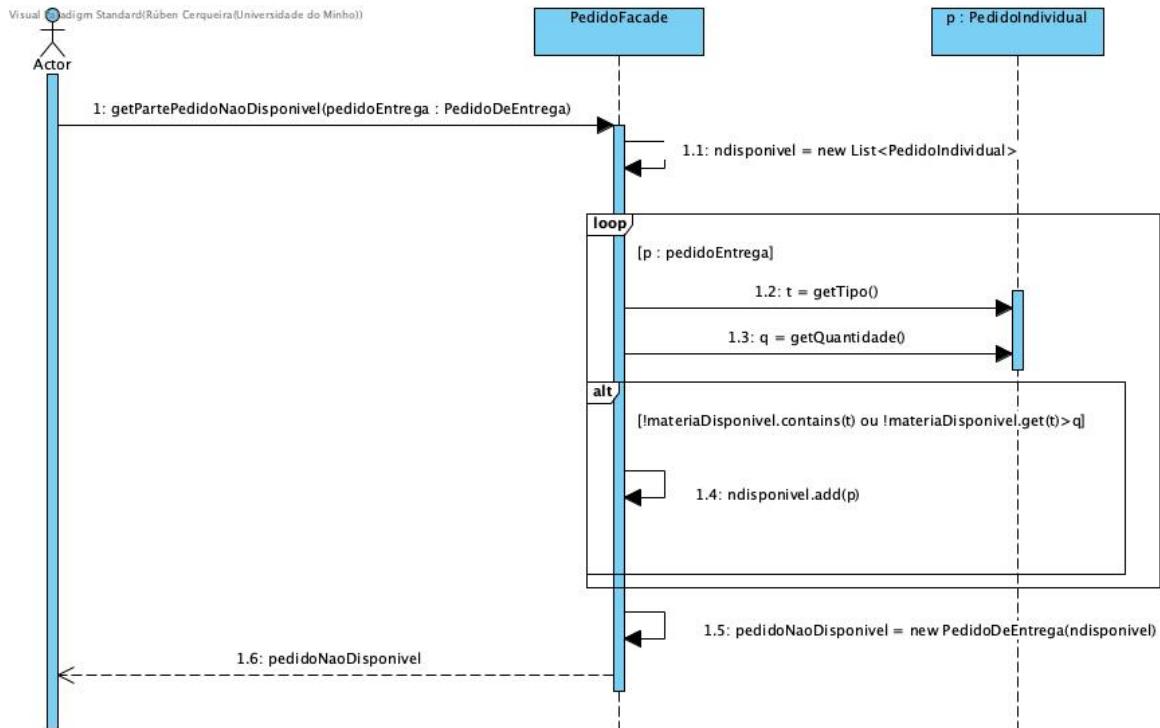


Figura C.14: Diagrama de Sequência do método `getPartePedidoNaoDisponivel`

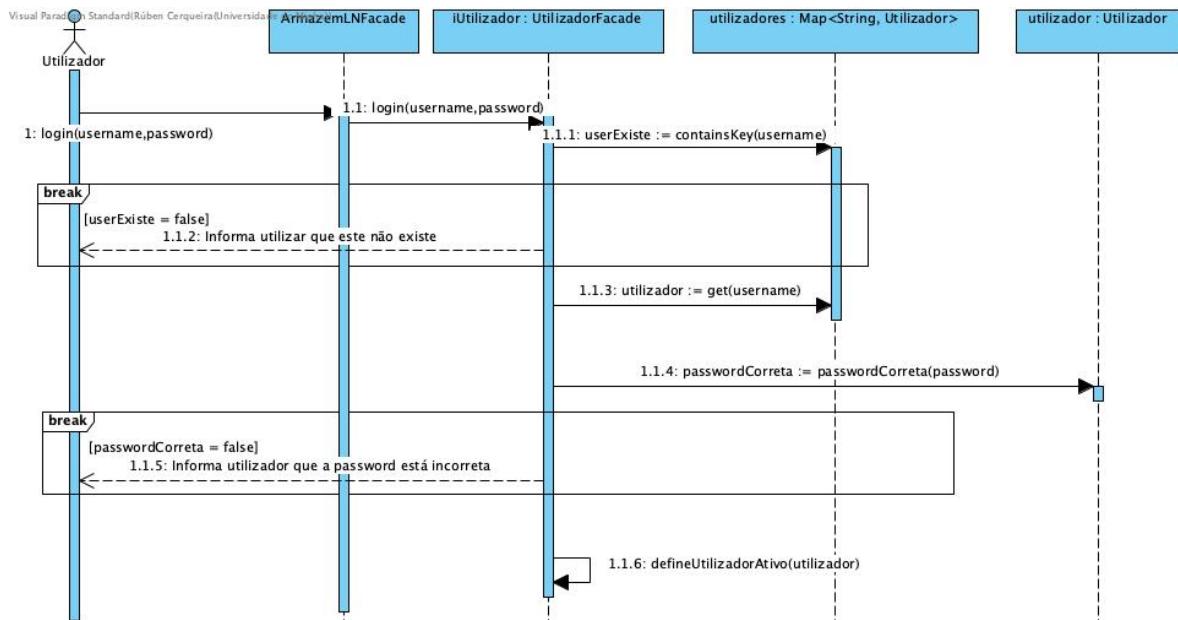


Figura C.15: Diagrama de Sequência do método login

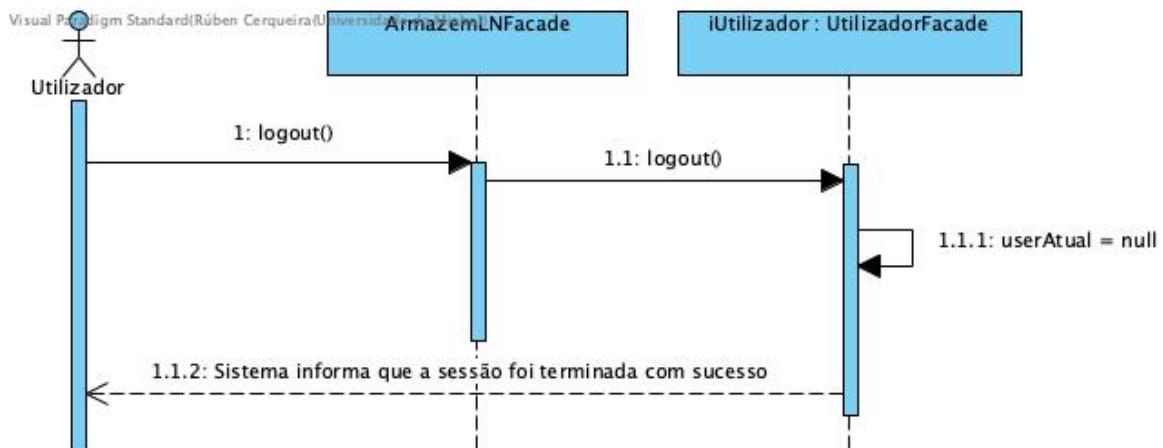


Figura C.16: Diagrama de Sequência do método logout

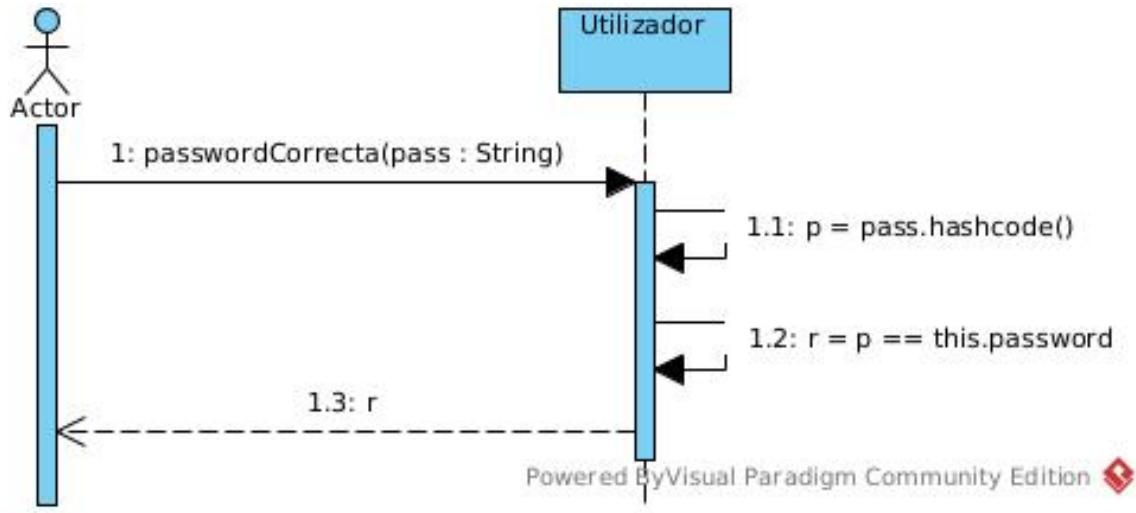


Figura C.17: Diagrama de Sequência do método passwordCorrecta

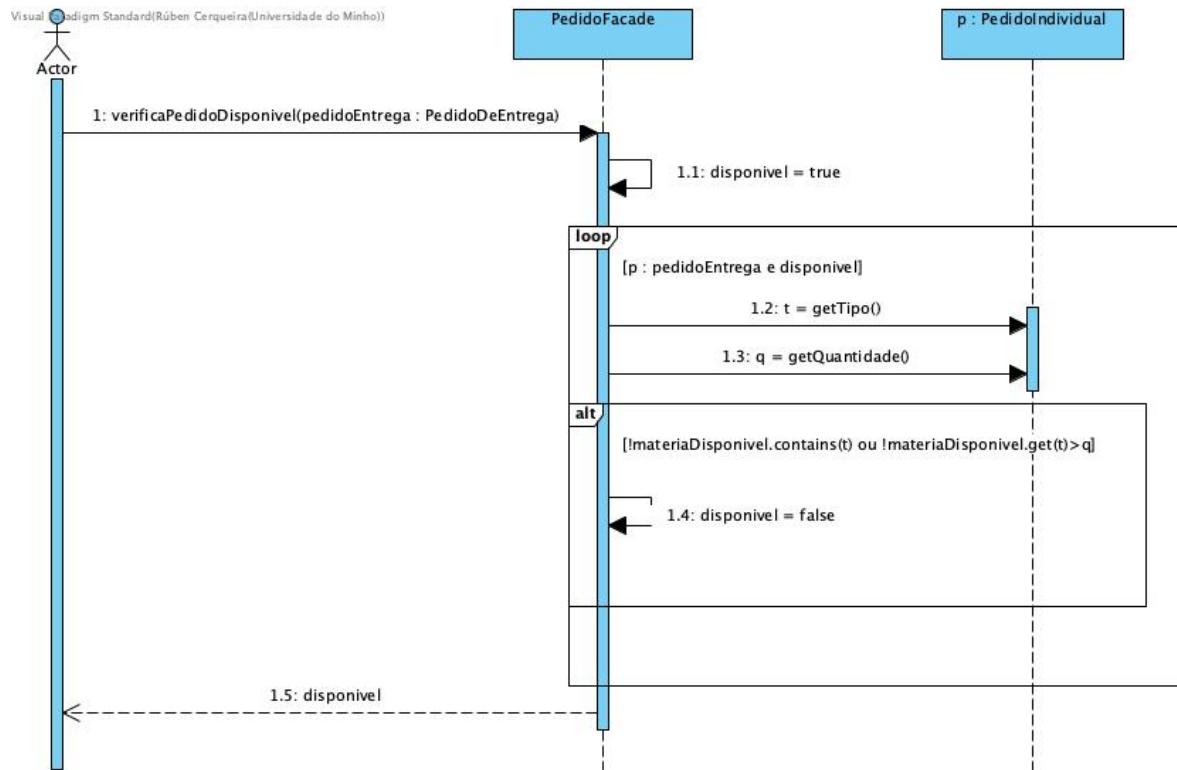


Figura C.18: Diagrama de Sequência do método verificaPedidoDisponivel