

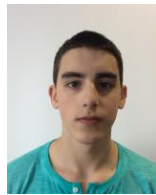
# Desenvolvimento de Sistemas de Software

## Mestrado Integrado em Engenharia Informática

Luís Capa  
A81960



Moisés Antunes  
A82263



Pedro Capa  
A83170



31 de Dezembro de 2018

## Conteúdo

<b>1</b>	<b>Introdução</b>	<b>2</b>
<b>2</b>	<b>Objetivos</b>	<b>2</b>
<b>3</b>	<b>Trabalho realizado</b>	<b>3</b>
3.1	Modelo de domínio . . . . .	3
3.2	Diagrama de use cases . . . . .	3
3.3	Especificações de use cases . . . . .	4
3.4	Protótipo da interface . . . . .	6
3.5	Máquinas de estado . . . . .	8
3.6	Diagrama de sequência de sistemas . . . . .	9
3.7	Diagrama de classes . . . . .	11
3.8	Diagramas de Implementação . . . . .	11
3.9	Diagrama de packages . . . . .	11
<b>4</b>	<b>Funcionamento da aplicação</b>	<b>11</b>
<b>5</b>	<b>Conclusão</b>	<b>13</b>

## 1 Introdução

Hoje em dia, os clientes das grandes marcas de automóveis têm a possibilidade de personalizar o carro da maneira como quiserem. Para tal, os clientes podem escolher peça a peça, escolher um pacote ou escolher configuração ótima em relação ao orçamento. O cliente não pode escolher todas as peças que quer, visto que há peças que são incompatíveis e outras que são obrigatórias. Os componentes na fábrica têm um stock, no caso de chegarem novas peças é necessário informar o sistema que chegaram novas peças e os carros que estão à espera dessas peças voltam para a fila de produção. Como este modelo de compra está a ser muito usado, o trabalho de DSS deste ano consistia em modelar o problema, recorrendo à linguagem de modelação UML, para criar a interface gráfica da aplicação e para criar as classes deste projeto, foi recorrido à linguagem de programação Java. O sistema de base de dados utilizado foi o MySQL.

## 2 Objetivos

Esta unidade curricular tem o objetivo de mudar a forma como nós programamos, pois até agora sempre que recebíamos um projeto o primeiro pensamento era escrever código. Para isso usou-se a linguagem UML para preparar a escrita do código de forma a cometer menos erros quer na criação de classes e nas suas variáveis de instância quer ter uma representação gráfica do sistema a ser desenvolvido. Este relatório tem o objetivo de clarificar o trabalho realizado no geral, bem como as suas partes.

### 3 Trabalho realizado

Para este trabalho foram criados alguns modelos UML como o modelo de domínio, UseCase, os diagramas de estado, os diagramas de sequência, diagramas de classes e os diagramas de packages.

#### 3.1 Modelo de domínio

Inicialmente, no "Visual Paradigm" foi criado um modelo de domínio que mostra os principais conceitos e relações entre entidades criadas para este problema, a título de exemplo, foram consideradas algumas relações entre os componentes do carro.

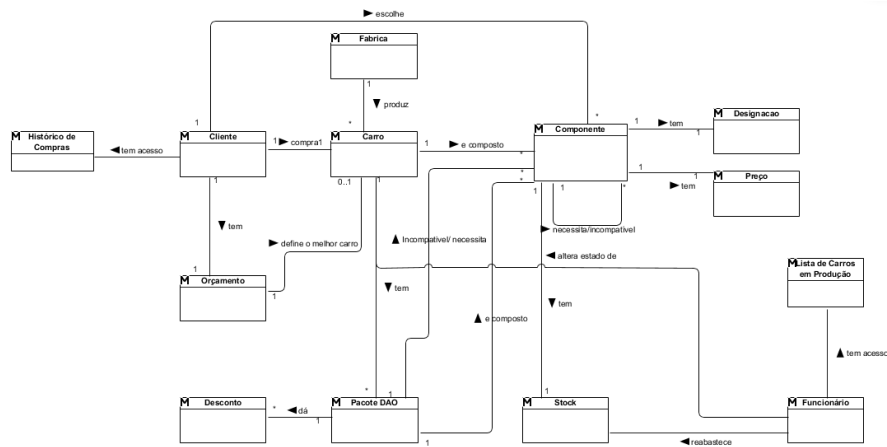


Figura 1: Modelo de domínio

#### 3.2 Diagrama de use cases

De seguida, no mesmo programa foi criado o modelo de use cases. Este modelo mostra as principais interações entre os atores do sistema com o próprio sistema. De notar que só foram criados os use cases que achamos absolutamente necessário para cumprir alguns requisitos pré-estabelecidos. Tendo em conta que o use case Comprar Carro iria ficar muito extenso, foi decidido que era melhor dividi-lo em quatro, o Comprar Carro, Escolher Pacote, Escolher Especificações e o Escolher Configuração Ótima. Para cada use case foi feita uma descrição detalhada da interação entre o ator e o sistema. Neste sistema foram considerados dois atores, o cliente que faria a compra online e o funcionário que utilizaria a aplicação na fábrica.

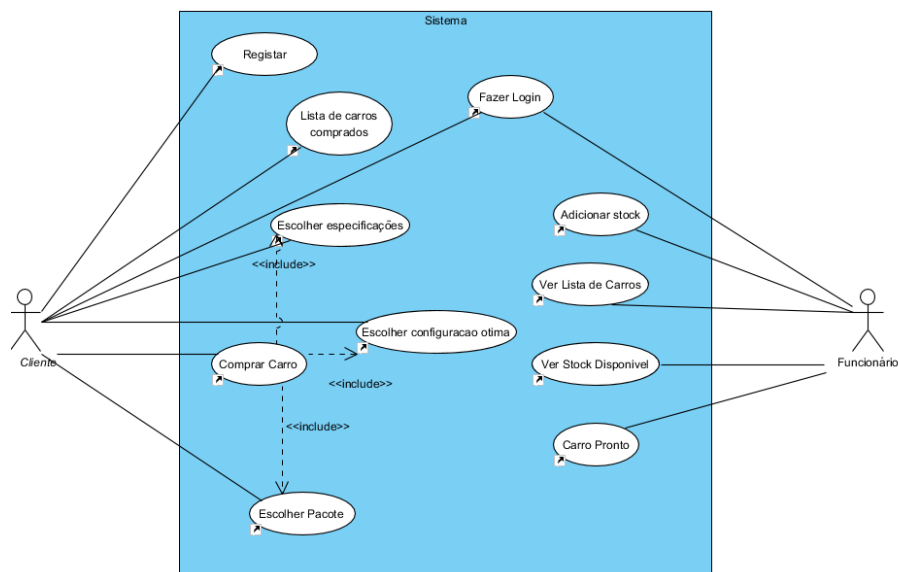


Figura 2: Diagrama de Use Cases

### 3.3 Especificações de use cases

Para cada use case, foi criado um modelo no "Excel", que tinha a interação detalhada entre o ator do sistema e o sistema, no qual representava o cenário normal da interação para esse use case, mas também mostrava alternativas e exceções a este. Este também indicavam as pré e pós condições do use case e qual o ator para esse use case em particular. As seguintes imagens são alguns exemplos da especificação de cada use case.

Use Case:	Stock Disponível	
Actor:	Funcionário	
Pré condição:	Actor já fez Login	
Pós condição:		
Cenário Normal	Actor input	System response
		1. Mostra opções
	2. Escolhe opção de ver Stock Disponível	
		3. Obtém stock disponível
		4. Mostra stock disponível

Figura 3: Descrição do use case Ver Stock Disponível

Use Case:	Login	
Actor:	Cliente, Funcionário	
Pré condição:	Estar registado no sistema	
Pós condição:	Ter acesso às suas informações	
Cenário Normal	Actor input	System response
		1. Apresenta menu inicial
	2. Indica que pretende fazer login	
		3. Pede credenciais
	4. Coloca credenciais	
		5. Valida credenciais
Exceção 1 [Dados inválidos] (Passo 5)		6. Informa que entrou com sucesso
		5.1. Informa que as credenciais são inválidas

Figura 4: Descrição do use case Login

Use Case:	Lista de Carros Comprados	
Actor:	Cliente	
Pré condição:	Registado no Sistema	
Pós condição:	Verificou lista de carros comprados	
Cenário Normal	Actor input	System response
		1. Mostra Menu de opções
	2. Seleciona historico de compras	
		3. Obtém lista de carros comprados
		4. Mostra lista de carros comprados

Figura 5: Descrição do use case Ver Lista de Carros Produção

Use Case:	Comprar Carro	
Actor:	Cliente	
Pré condição:	Estar autenticado no sistema	
Pós condição:	Carro encomendado	
Cenário Normal	Actor input	System response
		1. Mostra opções
	2. Indica que quer comprar carro	
		3. Obtém modelos disponíveis
		4. Mostra todos os modelos disponíveis
	5. Seleciona modelo que pretende comprar	
		6. Regista modelo escolhido
	8. Escolhe configuração ótima	
		9.<<include>> Escolher configuração ótima
Alternativa 1 [Escolher Pacote] (passo 8)	8.1 Escolhe Pacote	
		8.2 <<include>> Escolhe Pacote
Alternativa 2 [Escolher Especificações] (passo 8)	8.1 Escolhe Especificações	
		8.2 <<include>> Escolher Especificações

Figura 6: Descrição do use case Comprar Carro, que acaba por se dividir em mais três

### 3.4 Protótipo da interface

Tendo em conta os use cases que foram criados, foi pensado uma forma dos utilizadores interagirem com o sistema. Assim, foram feitas alguns protótipos para cada use case. Sempre que um utilizador realizava uma ação inválida aparecia uma nova janela no ecrã que indicava o motivo do aviso, por exemplo, se um utilizador se tentasse autenticar a aplicação mostrava uma janela que indicava que não se conseguiu autenticar. Quando fazia uma ação válida abria uma nova página e fechava a anterior, exceto se fosse o menu principal.



Figura 7: Menu Inicial

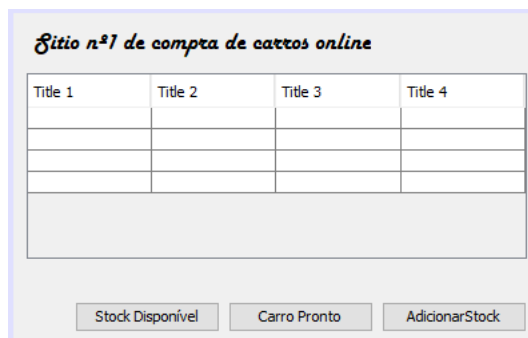


Figura 8: Menu principal do funcionário

**Identificador da peça**

**Quantidade**

Cancelar Confirmar

Figura 9: Janela na qual o funcionário vai adicionar stock

**Bem-vindo ao sitio n.º1 de compra de carros online**

Nome	Ano	Estado

Comprar Carro Historico de Compras

Figura 10: Menu principal do cliente

**Selecione a melhor opção para si**

☐ Model T

☐ Boss 429 Mustang

☐ Boss 302 Mustang

☐ Thunderbird

☐ Indigo Concept

☐ Edge

Figura 11: Menu no qual o cliente vai comprar o carro, começando por escolher o modelo e depois a forma da compra

Este sistema não foi totalmente fiável, porque à medida que o trabalho foi desenvolvido estes modelos foram um pouco alterados e ainda foi acrescentado um menu.

### 3.5 Máquinas de estado

No desenvolvimento da aplicação foram consideradas duas entidades que teriam diferentes estados, o carro e a peça. Para descrever as diferentes fases destas entidades foi criado para cada uma um modelo de estado. A figura 12 e a figura 13, representam, respetivamente o estado, quer do carro, quer da peça.

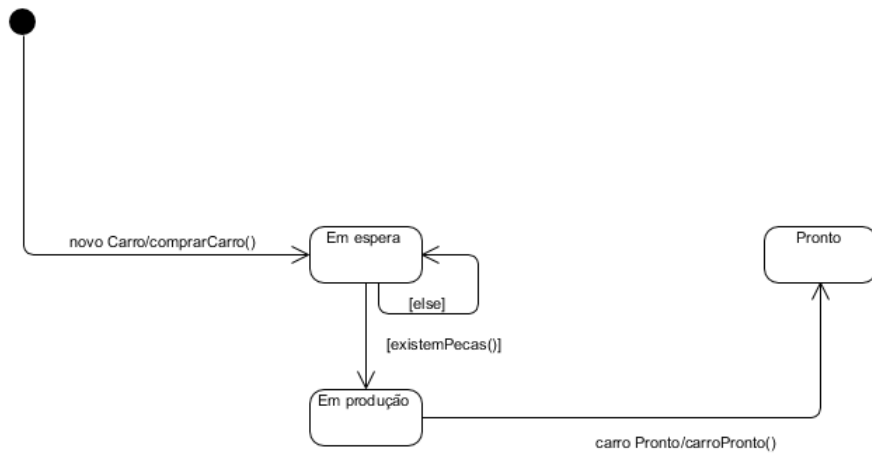


Figura 12: Máquina de estados da entidade Carro

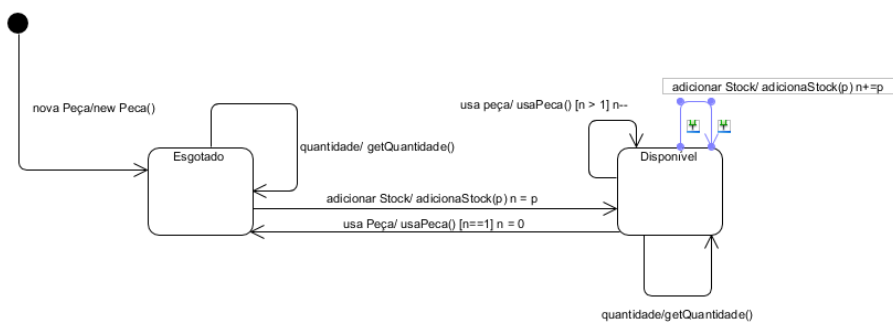


Figura 13: Máquina de estados da entidade Peça



### 3.6 Diagrama de sequência de sistemas

Para se obter uma melhor percepção da interação entre o cliente e o sistema foram criados os diagrama de sequência de sistema para cada use case, baseados nas descrições de cada um, dando uma melhor ideia da interação entre o ator e o sistema. As figuras abaixo, são alguns exemplos de DSS baseados na especificação dos use cases.

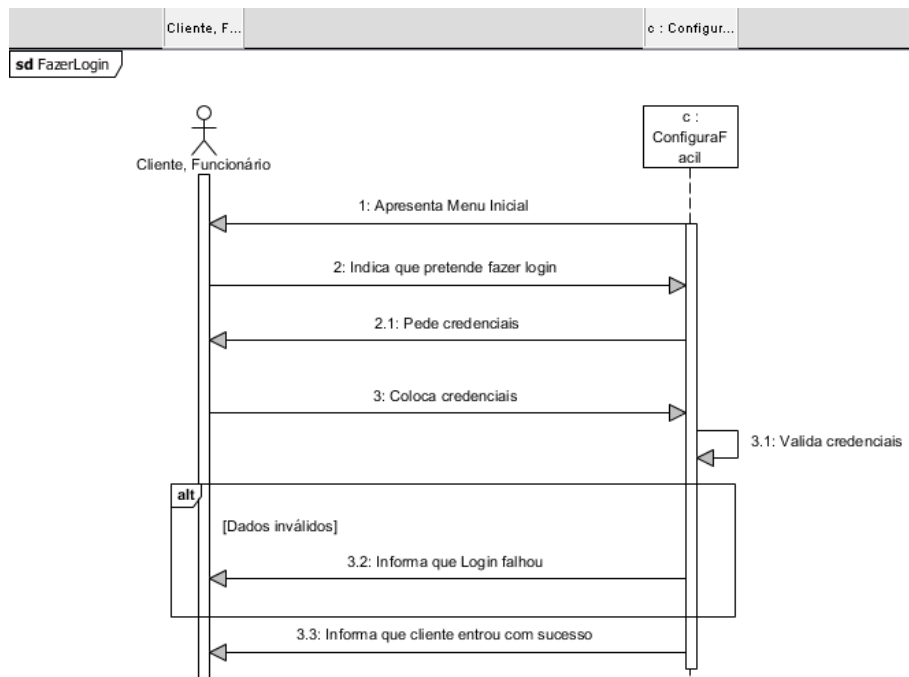


Figura 14: DSS do use case Fazer Login

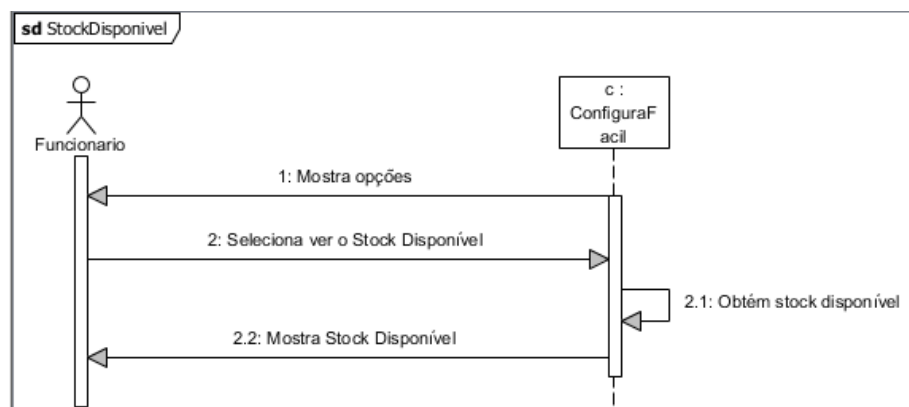


Figura 15: DSS do use case Stock Disponível

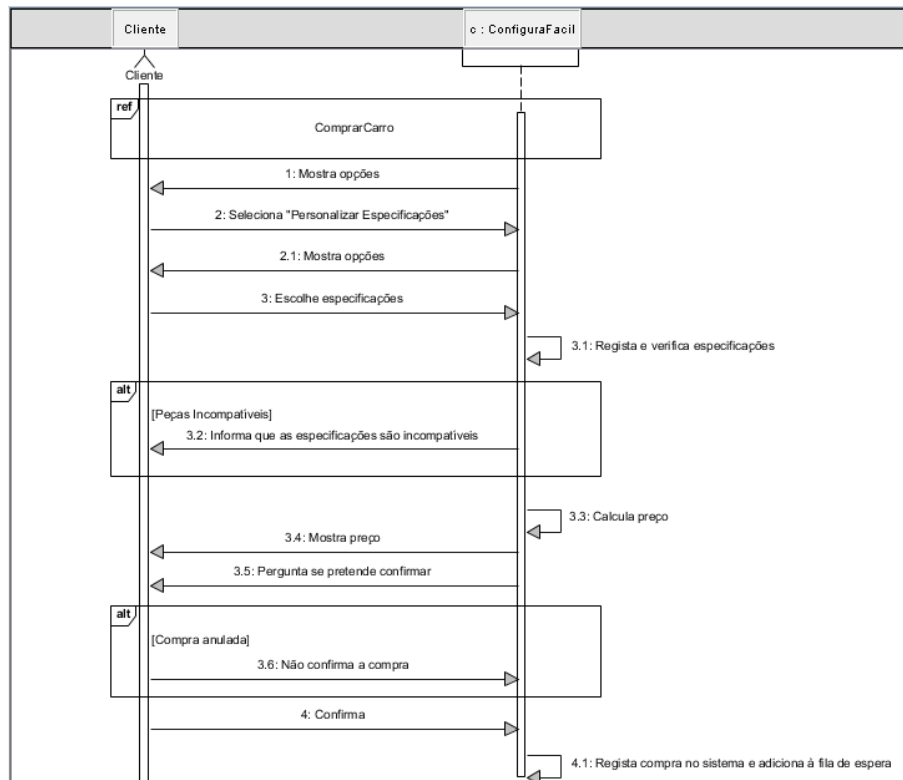


Figura 16: DSS do use case Escolher Especificações

Inicialmente foram usadas as descrições dos use cases criados em "Excel", para criar um modelo simples, mas em seguida foram criados mais alguns DSS e em cada um era aumentado o detalhe. Em segundo lugar, foram criados os DSS que além de representar a interação do sistema com o utilizador, também representava a relação entre a interface do sistema e a camada de negócio deste.

Em seguida foram criados os DSS de subsistemas, que, basicamente, representa as várias entidades no sistema e como cada use case interage com essas entidades, por exemplo o use case "Adicionar Stock" relacionava-se com o subsistema da peça e do carro.

### 3.7 Diagrama de classes

Para os DSS criados foi criado um diagrama de classes, que indicava as classes que eram precisas criar, as variáveis de instância para cada classe e todos os métodos que eram necessários implementar. Não foram criados diagramas de classes para todos os DSS, pois os use cases do "Escolher configuração ótima", "Escolher Especificações" e o "Escolher Pacote" foram todos incluídos no diagrama de classes do "Comprar Carro", pois os métodos nestes DSS eram quase sempre os mesmos e achamos que não havia necessidade de criar diagramas de classes para cada um. O mesmo aconteceu para os diagramas de classes para os DSS que tinham os DAO. No fim de todos os diagramas de classes serem feitos, foram todos juntados num só, em que continha todas as classes, variáveis de instância e métodos.

### 3.8 Diagramas de Implementação

De seguida, foram criados DSS para especificar os métodos que eram utilizados nos DSS anteriores. A partir de alguns dos últimos DSS, aqueles que tinham de aceder à base de dados, foram criados uns DSS, em que as listas e os conjuntos de entidades eram substituídas por DAO, classes que acediam à base de dados, por exemplo no método `getListaCarrosComprados()`, antes o sistema ia a um map buscar os carros que eram do cliente, mas no último DSS esse map era substituído pela classe `CarroDAO`, que continha um método para trazer da base de dados as especificações dos carros que o cliente comprou.

### 3.9 Diagrama de packages

Em seguida, os diagramas de classes e o modelo de domínio foram divididos em packages. Além destes diagramas de packages, do mesmo modo foi criado um diagrama de packages para demonstrar as diferentes camadas aplicacionais do sistema. Esta foi dividida em três partes, a Interface, a camada de negócio e a base de dados. Na camada de negócio estava contida as várias classes do sistema.

## 4 Funcionamento da aplicação

Como já foi referido acima há dois tipos de utilizadores, os clientes e os funcionários. Os funcionários são responsáveis por colocar um carro em produção pronto e adicionar peças na fábrica. Quando são adicionadas peças pelos funcionários, o sistema verifica se há algum carro à espera dessa peça e põe essa peça como colocada no carro. O cliente tem acesso à lista de carros que comprou, pode comprar um carro de três formas diferentes, escolher um pacote, escolher a melhor configuração com um determinado orçamento ou personalizar

por completo o carro. Na compra do carro, o cliente escolhe primeiro qual o modelo do carro e só em seguida escolhe a forma de compra. Se um cliente escolher um pacote, este não tem a opção de alterar as peças obrigatórias, neste caso, o motor, a cor, jantes e estofos, no entanto, pode escolher todas as extras que quiser. Ao escolher esta opção o cliente tem um desconto associado a cada pacote. Se o cliente escolher personalizar tem a possibilidade de escolher todas as peças que quiser, desde que não tenha peças incompatíveis ou não tenha peças obrigatórias em falta. Ao escolher a configuração ótima não há garantia que tenha a melhor configuração dado o orçamento disponível. Nesta opção o sistema vai tentar encontrar o melhor pacote para, ou seja, o pacote mais caro para o dado modelo e orçamento que escolheu. Se ainda houver margem o sistema vai adicionar todos os extras que puder, começando sempre por adicionar os mais caros. Este algoritmo não garante que seja escolhida a melhor configuração, pois não verifica todas as diferentes.

## 5 Conclusão

No final, acreditamos que o trabalho que desenvolvemos até agora demonstra bem esse equilíbrio entre personalização e facilidade de uso que procuramos, que são apenas detalhes, mas é neles que se destaca uns projetos dos outros. Também achamos que o essencial foi feito apropriadamente e que não há falta de alguma funcionalidade que seja absolutamente necessário, sendo que consideramos o trabalho até agora tanto funcional como cómodo para o utilizador. Como não tínhamos muita prática em modelação, à medida que o projeto era desenvolvido verificamos que eram cometidos alguns erros na modelação, que nos obrigava a reformular alguns modelos, retardando o desenvolvimento do projeto. Foi notado que se passou menos tempo a corrigir erros no código, mas também foram criados menos métodos e que só foram criados os métodos que eram necessários no desenvolvimento do projeto, comparando com outros projetos, que por vezes eram criados métodos que nunca eram utilizados.