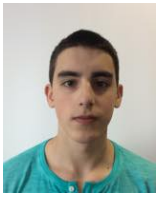
Desenvolvimento de Sistemas de Software

Mestrado Integrado em Engenharia Informática

Luís Capa
A81960

Moisés Antunes
A82263



Pedro Capa
A83170

28 de Dezembro de 2018

Conteúdo

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.

2 Objetivos

Esta unidade curricular tem o objetivo de mudar a forma como nós programamos, pois até agora sempre que recebiamos 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 dominio, UseCase, os diagramas de estado, os diagramas de sequência, diagramas de classes e os diagramas de packages.

Inicialmente, no "Visual Paradigm"foi criado um modelo de domínio que mostra as prinicipais relações entre entidades criadas para este problema, a título de exemplo, foram consideradas algumas relações entre os componentes do carro. 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 própio sistema.De notar que só foram criados os use cases que achamos absolutamente necessário para cumprir alguns requisitos pré-estabelecidos. Para cada use case foi feita uma descrição detalhada da iteração 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.

O cliente tem os seguintes use cases:

- Registar;
- Comprar carro;
- Ver lista de carros comprados;
- Escolher pacote;
- Escolher especificações;
- Escolher configuração ótima;
- Login

O Funcionário tem os seguintes use cases:

- Login
- Adicionar stock
- Ver stock disponível
- Carro pronto
- Ver lista de carros

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.

Para o carro foram considerados os seguintes estados:

Em espera Quando algum componente do carro está sem stock;

Em produção No caso de todos os componentes para o carro existirem;

Pronto Ao fim do carro ser produzido;

A peça ou componente tinham os seguintes estados:

Esgotado Se não houver em stock esta peça;

Disponível No caso de haver em stock esta peça

Para se obter uma melhor perceção da interação entre o cliente e o sistema foram criados os DSS para cada use case. 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 sguida 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. De seguida, foram criados DSS para especificar os métodos que eram utilizados nos DSS anteriores. A partir de alguns dos útlimos DSS, aqueles que tinham de aceder à base de dados, foram criados uns DSS, em que as listas e os conjuntos de entidades eram subsitituídas 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.

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.

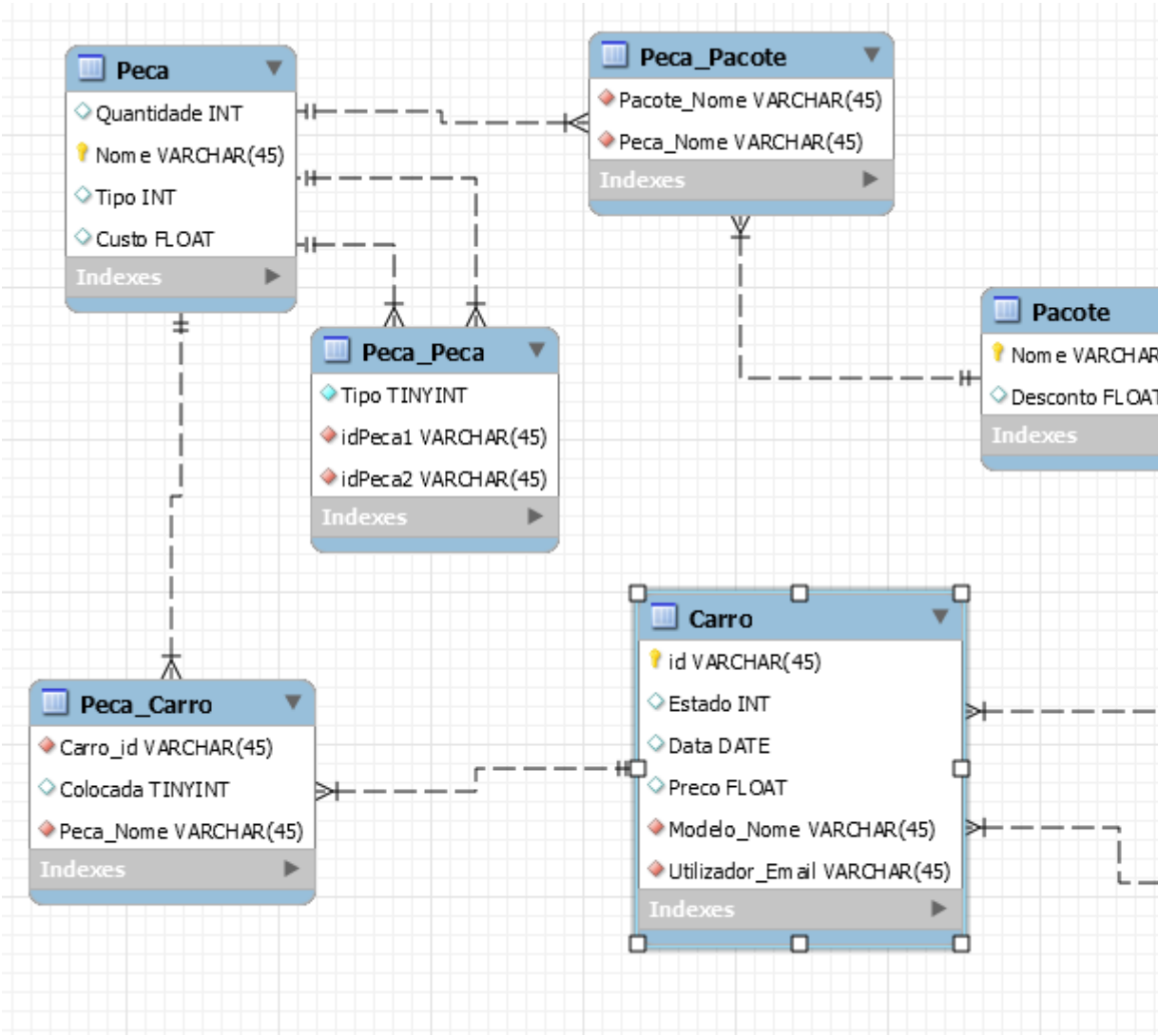
Em seguida, os diagramas de classes e o modelo de domínio foram divididos em packages. Além destes diagramas de packages, do mesmo modo foi criado um diagrama de packages para demonstrar as diferentes camadas aplicacionais do sistema. Esta foi dividida em três partes, a Interface, a camada de negócio

e a base de dados. Na camada de negócio estava contida as várias classes do sistema.

Finalmente, foi começado a implementar a aplicação, na qual o trabalho foi dividido em três partes, a interface, a base de dados e a camada de negócio. Cada uma destas partes foi feita por cada um dos elementos do grupo. No entanto, quando foram criadas as classes DAO foi notado que na altura de carregar os dados, por exemplo de uma peça podia ser carregado para memória muitas peças que nem eram precisas, uma vez que uma peça tem uma lista de peças proibidas e uma lista de peças obrigatórias, ou seja para carregar uma peça da base de dados tinha-se de levar por arrasto muitas peças. Este problema também se verificava com os carros, por causa da lista de peças que um carro tem. Depois deste problema foi dicidido alterar algumas classes, que continham listas, como por exemplo a peça em vez de ter uma lista de peças passou a ter uma lista de string.

A interface foi criada no "NetBeans", uma vez que é mais fácil de implementar a interface gráfica, porque tem um visualisador de classes de interfaces e gera automaticamente o código dos componentes do swing, api de interfaces usada. A interface foi modelada de acordo com os Use Cases do projeto, uma vez que este indica que opções cada ator realiza no sistema.

A base de dados utilizada foi "MySQL", porque já tinha sido explorada em outras UC, mas também por ser uma base de dados relacional.



Por fim, a base de dados foi povoada e foram feitos alguns testes, que foi útil para corrigir pequenos erros no código.

4 Análise crítica do trabalho realizado

Vários "Use Cases" foram criados para garantir acesso a um sistema que permita tanto aos clientes como funcionários proceder à compra, venda e outras operações relativas aos automóveis da empresa. O nosso objetivo principal durante a criação destes "Use Cases" foi garantir que haveria várias opções de customização de carros e que os utilizadores do sistema teriam acesso aos dados que lhes diziam respeito sem, no entanto, exagerar na quantidade de opções disponíveis, que deixaria o sistema demasiado complicado de usar, especialmente para clientes não acostumados com algo do género.

Nos modelos de estado consideramos que os principais métodos e interações com as entidades foram considerados, apesar de eventualmente haver algum em falta, por exemplo não foi considerado o caso de alguma peça não ser mais produzida pelos fornecedores da fabrica.

Para o modelo de domínio acreditamos que os principais conceitos e relações entre entidades foram inseridas neste modelo.

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 tinhamos 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.