Relatório do Projeto

Sistema de Gestão de um Stand de Automóveis

Realizado por:

- Pedro Daniel Fernandes Rei (26013)

Índice

Introdução

O presente relatório descreve o desenvolvimento de um Sistema de Gestão para um Stand de Automóveis, em linguagem C#, com base nos conteúdos apreendidos nas aulas da unidade curricular Programação Orientada a objetos, da licenciatura Engenharia em Sistemas Informáticos.

O objetivo principal deste projeto é otimizar e automatizar os processos que estão relacionados á administração e vendas de automóveis, de modo a proporcionar uma gestão eficiente e melhorar a experiência tanto para os respetivos clientes do stand, como para a equipa interna do stand.

A realização do projeto foi feita sobre o paradigma da programação orientada a objetos, onde o foco foi sobretudo nas classes, os seus respetivos objetos, e as heranças que estabelecem relação entre as diversas classes.

Desta forma o projeto não apenas pretende apenas implementar a otimização dos processos de um stand de automóveis, mas sim como um exemplo prático dos princípios apreendidos durante as aulas, de modo a ganhar competências para situações do mundo real.

Objetivos do Projeto

Os objetivos definidos e desenvolvidos para a realização deste projeto realizado na linguagem de programação C# foram os seguintes:

- Consolidar conceitos basilares do Paradigma Orientado a Objetos;
- Analisar problemas reais;
- Desenvolver capacidades de programação em C#;
- Potenciar a experiência no desenvolvimento de software;
- Assimilar o conteúdo da Unidade Curricular Programação Orientada a Objetos;
- Otimizar e automatizar os processos que estão relacionados á administração e vendas de automóveis

Primeira Fase

A primeira fase consiste em identificar as classes no projeto que vai ser desenvolvido e implementar estas. Neste caso foram identificadas as seguintes:

- Pessoa
- Automóvel
- Trabalhador
- Dono
- Cliente
- Clientes
- Carro
- Carros
- Mota
- Motas
- Administrador
- Administradores
- Vendedor
- Vendedores
- Limpador
- Limpadores

Diagrama de Classes (Completo)

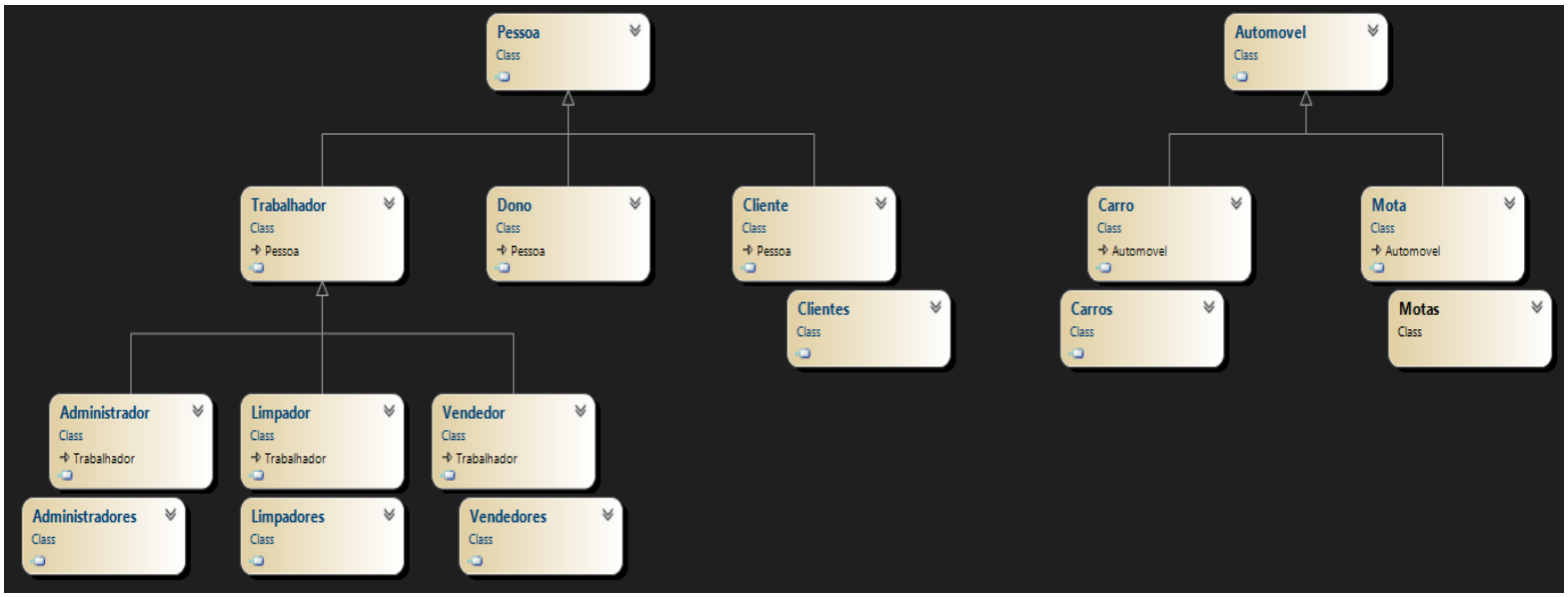

Figura 1- Diagrama de Classes do Sistema de Gestão de um Stand

Na figura 1, temos uma imagem do Diagrama de Classes do projeto na sua totalidade, onde vemos representado como classe Pai as seguintes classes: classe Pessoa, e a classe Automóvel.

As classes Trabalhador, Dono e Cliente herdam da classe Pessoa, tendo assim os atributos e métodos desta. A classe Trabalhador também vai ser classe Pai das classes Administrador, Limpador e Vendedor, tendo assim também os atributos e métodos desta. As classes Carro, e Mota herdam da classe Automóvel, tendo assim os atributos e métodos desta classe.

Para armazenar a informação, este processo foi feito através das estruturas de dados Array, que basicamente está implementada nas classes Administradores, Limpadores, Vendedores, Carros e Motas.

Diagrama de Classes (secção Pessoas)

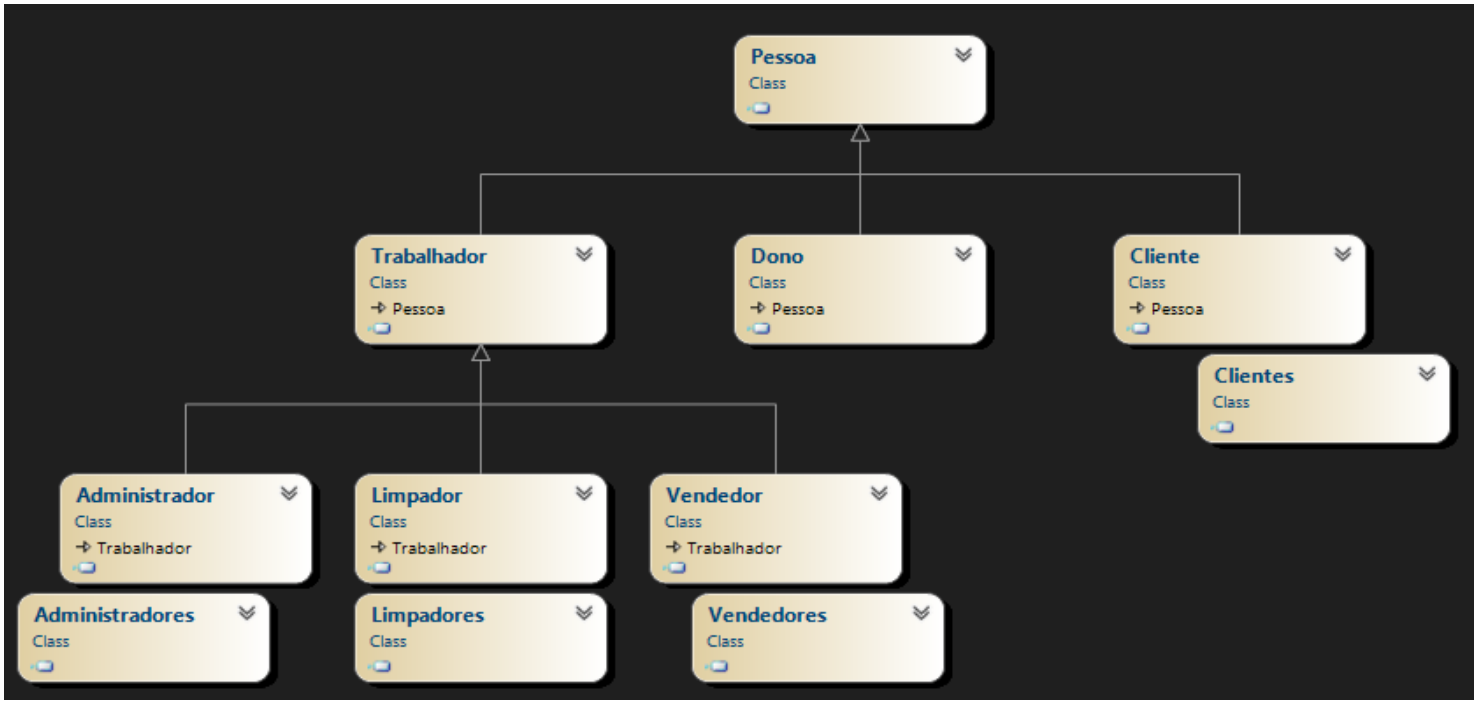

Figura 2 – Diagrama de classes da secção Pessoa

Na figura 2, temos somente a secção Pessoa do Diagrama de Classes, basicamente a classe Pessoa vai ter como atributos nome, idade, data nascimento, sexo, morada. A classe Trabalhador herda os atributos de Pessoa, e tem alem destes têm os atributos código trabalhador, anos de serviço e stand. A classe Dono herda os atributos da classe Pessoa e além deste têm os atributos código de dono. A classe Cliente herda os atributos de Pessoa e além deste têm os atributos NIF, código cliente, contacto, email,

número contribuinte e saldo. Depois temos as classes Administrador, Limpador, Vendedor que vão herdar os atributos de Trabalhador e além destes vão ter respetivamente os seus atributos, id administrador, id limpador, id vendedor e salário, nível cargo e funções.

Diagrama Automóvel (secção Automóvel)

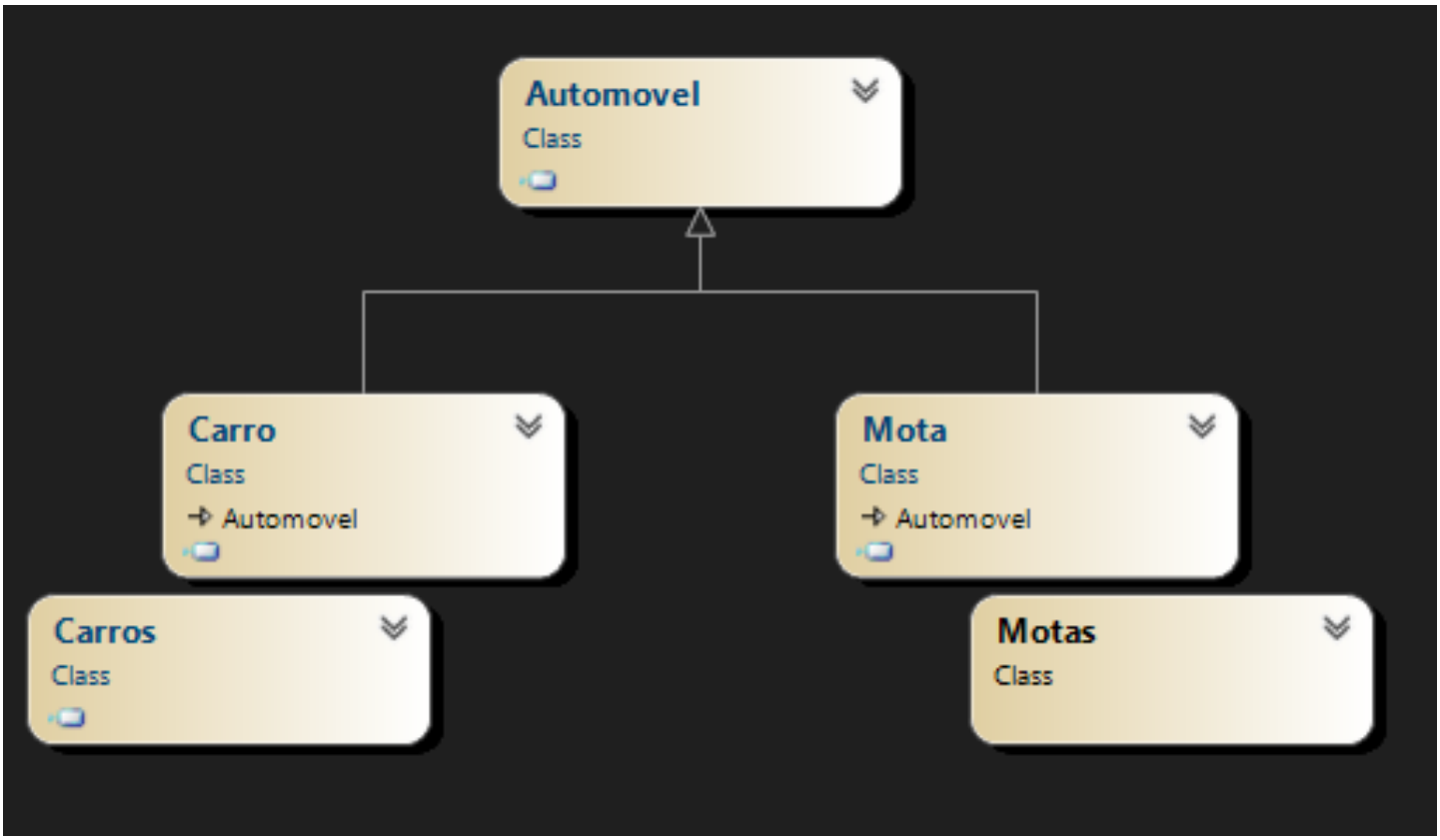

Figura 3 – Diagrama de classes da secção Automóvel

Na figura 3, temos somente a secção Automóvel do Diagrama de classes, onde visualizamos a classe Pai que é Automóvel, que contém os atributos data de fabrico, marca, matrícula, modelo, preço. Depois temos a classe Carro que herda de Automóvel, contendo os atributos desta e além desses os atributos código carro, lugares, consumo, cor e cavalos. De seguida a classe mota que vai ter os atributos código mota, consumo, cor. A classe Carros vai conter uma array do tipo Carro que vai armazenar a informação dos carros do stand. A classe Motas vai ter um array do tipo Mota que vai armazenar a informação das motas do stand.

Implementação de Classe

As classes Pessoa, Automóvel, Cliente, Dono, Trabalhador, Vendedor, Limpador, Administrador, Carro, Mota apesar de terem diferentes atributos e diferentes funcionalidades, são estruturadas da mesma forma. Os seus componentes são os seguintes:

- **Atributos**: Estes servem para armazenar informações específicas de cada classe (ou no caso de herança herdar informações da classe base). Cada atributo representa uma característica ou propriedade da entidade da classe.

- **Construtor**: São definidos dois construtores, sendo o primeiro um construtor padrão sem parâmetros que serve para os atributos serem inicializados com valores padrão. O segundo é um construtor personalizado que aceita parâmetros para inicialização personalizada.

- **Propriedades**: As propriedades são métodos de acesso que permitem a modificação e a leitura de atributos. Cada propriedade é associada a um atributo específico e é definido por "get" (para leitura) e por "set" (para modificação).

- **Overrides**: A principal finalidade é permitir substituir o comportamento de um método na classe derivada, adaptando às necessidades específicas. Os métodos utilizados no override foram o "Equals" que é utilizado para comparar dois objetos quanto à igualdade (no caso deste método não existe uma implementação específica) e foi utilizado também o "ToString" que retorna uma string formatada com informações relevantes sobre uma instância da classe.

Nas classes Vendedores, Limpadores, Administradores, Clientes, Carros e Motas, que contém os arrays onde vai ser armazenada a informação, temos a seguinte estrutura:

- **Construtor:** Também utiliza um construtor padrão sem parâmetros, mas que utiliza os arrays atribuídos da classe inicializando um novo array de objetos da classe com um tamanho definido.

- **Propriedades:** Servem da mesma forma para a leitura e modificação de atributos, mas no caso deste tipo de classes é utilizado uma propriedade para a leitura e acesso do array da classe e outra propriedade para modificar e aceder um atributo específico definido por cada classe.

- **Overrides:** Neste tipo de classes os overrides define 3 métodos para a manipulação e gestão, que são o método de Adicionar, Remover e Contar. Para além desses cada classe de manipulação de arrays tem assinaturas de métodos que são especificados nas respetivas interfaces.

Segunda Fase

Na segunda fase foram adicionadas e implementadas as classes no projeto as seguintes classes novas:

- Venda
- Vendas

Diagrama Classes

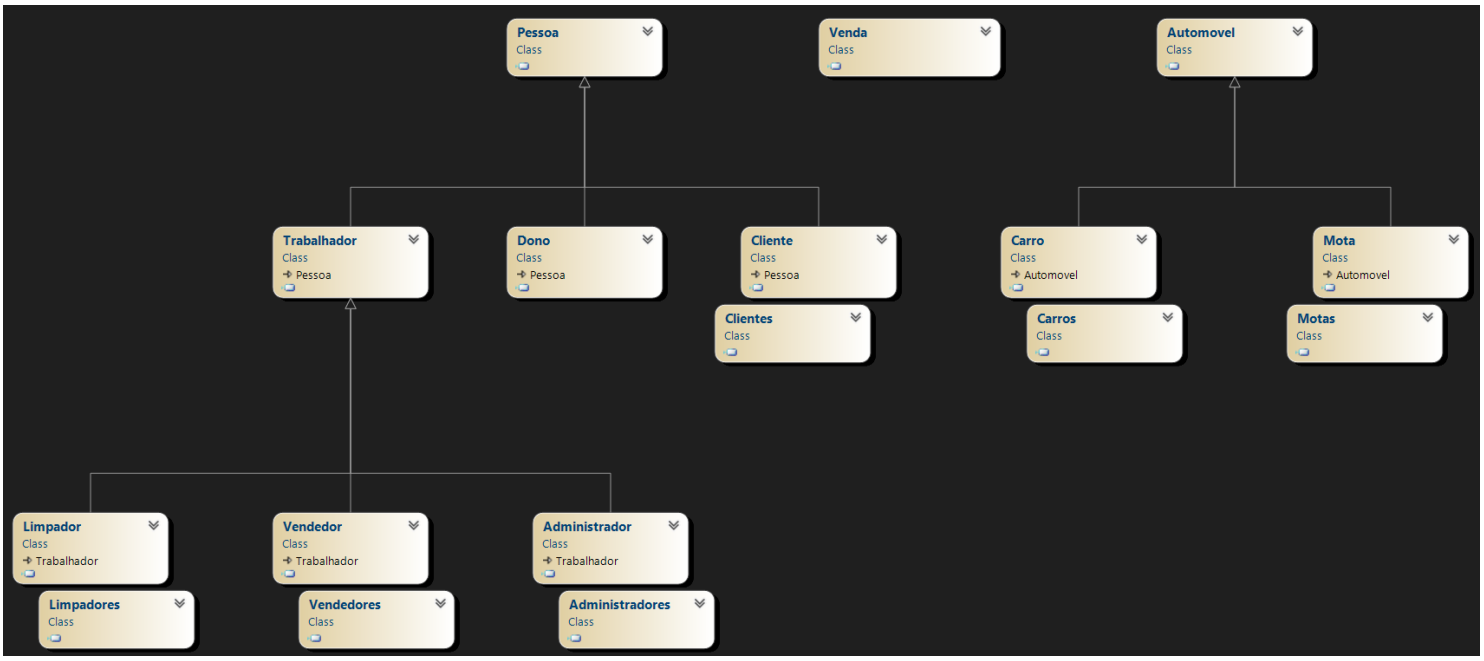

Figura 4 - Diagrama de Classes Completo

Arquitetura do Projeto (N-Tier)

Este projeto, principalmente nesta segunda fase passou por uma mudança na forma como se encontra organizado, passando a estar organizado em camadas.

Foi dividido em 4 camadas, em que o seu nome demonstra aquilo que representam. As camadas foram as seguintes: **Main**, onde se encontra o programa implementado; **BaseDados**, que contém as classes onde se encontram as listas que guardam os dados, e os seus métodos; **ObjetosNegocio**, que contém as classes onde estão definidos os diversos tipo de objetos que o projeto vai ter, e os seus atributos e métodos; **RegrasNegocio**, onde se encontram os métodos que são responsáveis por fazer a comunicação entre o backend e o frontend, neste caso a **BaseDados** e **Main**.

Este formato permite um melhor controlo da estrutura do programa, e permite uma melhor organização dos diversos componentes, facilitando o trabalho entre várias pessoas, o que não era o caso deste projeto.

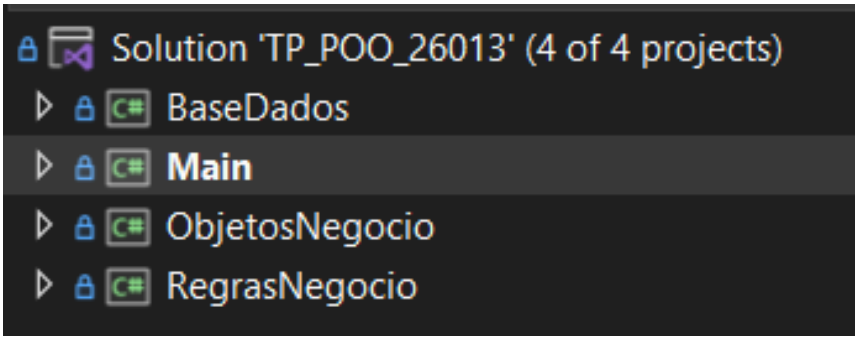

Figura 5 – Arquitetura em N-Tier

Estrutura de Dados

Nesta segunda fase a estrutura de dados que é responsável por armazenar os dados, foi trocada de arrays para listas de forma a implementar as regras definidas para esta fase.

Os dados dos objetos das classes Venda, Carro, Mota, Cliente, Vendedor, Administrador, Limpador foram armazenados nas listas deste tipo de objetos, na região **ATRIBUTOS** das seguintes respetivas classes: Vendas, Carros, Motas, Clientes, Vendedores, Administradores e Limpadores.

Nestas classes foram implementados ainda diversos métodos, como aqueles responsáveis por adicionar objetos á lista, remover objetos da lista, ordenar a lista por ordem crescente de código de identificação,

Além destes foram implementados dois métodos responsáveis por fazer a leitura dos dados dos ficheiros respetivos, e guardar os dados nestes ficheiros. De forma que no início do programa se tenha a informação toda necessária e no fim deste ela fique guardada.

Regras de Negócio

Nesta camada foram implementados diversos métodos que eram responsáveis por fazer a comunicação entre a **BaseDados** e **Main**. Nesta foram definidos de certa forma os filtros para se poder fazer diversas coisas, neste caso interagir com a base de dados, como por exemplo para se poder adicionar um cliente a lista de clientes este tinha de ter uma idade mínima de 18 anos.

Testes Unitários

Foram criados diversos testes unitários de modo a verificar a boa funcionalidade do programa, e fiabilidade deste.

Conclusão

Em conclusão, o desenvolvimento do Sistema de Gestão para o Stand de Automóveis em C# demonstra a aplicação prática dos princípios aprendidos na Programação Orientada a Objetos. Ao enfocar classes, objetos e heranças, foi conseguida criar uma solução eficiente que automatiza e aprimora os processos administrativos e de vendas. Esta experiência não apenas atende às necessidades específicas do stand, mas também proporciona habilidades transferíveis e uma base sólida para enfrentar desafios na área da informática, exemplificando a utilidade direta dos conceitos aprendidos em situações do mundo real.

Bibliografia

- Repositório GitHub das Aulas: https://github.com/luferIPCA/LESI-POO-2023-2024;
- Sebenta C# fornecida pelo docente;
- Youtube.