

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Graduação em Sistemas de Informação

Igor Palhares
Paulo Henrique

**TRABALHO FINAL INTERDISCIPLINAR DE PROGRAMAÇÃO ORIENTADA
A OBJETOS**

Belo Horizonte
2019

Igor Palhares Reis
Paulo Henrique Rocha Ribeiro

**TRABALHO FINAL INTERDISCIPLINAR DE PROGRAMAÇÃO ORIENTADA
A OBJETOS**

Trabalho apresentado à disciplina
Programação Orientada a Objetos, do curso
de Sistemas de Informação da Pontifícia
Universidade Católica de Minas Gerais.

Orientador: Prof. Paulo Cesar do Amaral
Pereira

Belo Horizonte
2019

RESUMO

Este projeto se refere a um sistema que verifique a pontuação na carteira de habilitação de usuários diversos a partir de multas, associação entre multas e veículos, possível suspensão de condutores.

Para a realização deste projeto, utilizamos diversos recursos práticos e teóricos desenvolvidos em salas de aula e em laboratório de programação orientada a objeto, além de pesquisas bibliográficas, consultas à fóruns, site do Detran e diversos sites relacionados ao tema.

A proposta é desenvolver um sistema mais realista possível baseado nas pesquisas realizadas pelo grupo e técnicas de programação orientado a objeto que tem como objetivo a utilização dos conceitos aprendidos em sala de aula, como herança, polimorfismo, coesão, modularização, entre outros e aplica-los na solução de um problema, neste caso, desenvolver um sistema de controle de veículos, motoristas e multas.

Após a finalização deste projeto, percebemos a importância e a melhoria que tivemos na criação de programas orientados a objeto, organizando melhor os códigos produzidos, facilitando manutenções e implementações futuras, além disso, no progresso que tivemos na administração e comunicação entre os membros do grupo.

Palavras-chave: Orientação a Objetos. Desenvolver um sistema. Aplicação prática da programação orientada a objeto.

Sumário

1. Introdução	5
1.1 Objetivos.....	5
1.2 Organização do Trabalho	6
1.3 Utilização do Programa.....	6
2. Desenvolvimento	19
2.1 Motoristas	19
2.1.1. Tipos de carteira	20
2.2 Veículos.....	22
2.2.1. Registrando um novo carro	22
2.2.2. Emplacando carro novo	22
2.3 Multas.....	23
2.4 Códigos e UML.....	25
2.4.2 UML	25
2.4.3 Códigos e Interfaces Gráficas.....	25
2.4.3.1 Códigos da classe Controle	25
2.4.3.3 Códigos da classe Motoristas	31
2.4.3.5 Códigos da classe Veículo	35
2.4.3.6 Códigos da classe Multa	38
2.4.3.8 Interfaces gráficas e seus códigos, começando pelo Menu Principal (frmPrincipal)	43
2.4.3.8.1 Código do frmPrincipal:	43
2.4.3.9 Opções para Motorista (frmOpcoesMotorista)	45
2.4.3.9.1 Código do frmOpcoesMotorista:	46
2.4.3.10 Opções para Motorista (frmIncluirMotorista).....	47
2.4.3.10.1 Código do frmIncluirMotorista:.....	47
2.4.3.11 Relatórios de Motoristas (frmRelatorioMotoristas):	52
2.4.3.11.1 Códigos do frmRelatorioMotoristas:.....	52
2.4.3.12 Opcoes Veículo (frmOpcoesVeiculo)	55
2.4.3.12.1 Código Opcoes Veículo (frmOpcoesVeiculo)	56
2.4.3.13 Incluir Veículo (frmIncluirVeiculo):	57
2.4.3.13.1 Código do frmIncluirVeiculo:.....	58
2.4.3.14 Excluir Veículo (frmExcluirVeiculo):	62
2.4.3.14.1 Códigos do frmExcluirVeiculo:.....	63
2.4.3.15 Relatórios Veículos (frmRelatorioVeiculos):.....	66
2.4.3.15.1 Códigos do frmRelatorioVeiculos:	66
2.4.3.16 Multas (frmOpcoesMulta)	70
2.4.3.16.1 Código Multas (frmOpcoesMulta):.....	70
2.4.3.17 Incluir Multa (frmIncluirMulta):	72
2.4.3.17.1 Código do frmIncluirMulta:	72
2.4.3.18 Gerenciar Multas (frmGerenciarMultas):	76

2.4.3.18.1 Código do frmGerenciarMultas:.....	76
2.4.3.19 Relatórios Multas (frmRelatorioMultas):	92
2.4.3.19.1 Códigos do frmRelatorioMultas:	92
3. CONCLUSÃO.....	107
REFERÊNCIAS.....	108

1. Introdução

De acordo com o tema escolhido, Gestão de condutores e veículos, foram realizadas diversas pesquisas e testes com diferentes ferramentas, com o objetivo de deixar o sistema mais próximo da nossa realidade.

Através das especificações ditas pelo nosso professor, criamos um sistema seguindo o que foi visto em sala de aula, relacionado a programação orientada a objeto.

O sistema consiste em gerenciar um sistema que verifique pontuação a carteira de habilitação de usuários diversos a partir de multas, associação entre multas e veículos, possível suspensão de condutores. Na tela inicial, será disponibilizado várias opções relacionados ao tema para que o usuário possa escolher, como: (botão de Incluir Motorista, Incluir Veículo, Incluir Multa, gerenciar multas, excluir veículo, relatórios e o botão de sair). Cada um desses botões será responsável para realizar alguma função e facilitar o gerenciamento do sistema.

1.1 Objetivos

Com o conhecimento obtido através das aulas, pesquisas sobre POO, aulas de Laboratório de POO, o Trabalho Interdisciplinar tem como objetivo desenvolver um sistema que, a partir de dados lidos de um arquivo, verifique pontuação na carteira de habilitação de usuários diversos a partir de multas, associação entre multas e veículos, possível suspensão de condutores, entre outros. Contendo diversos tipos de multas e para veículos e condutores.

1.2 Organização do Trabalho

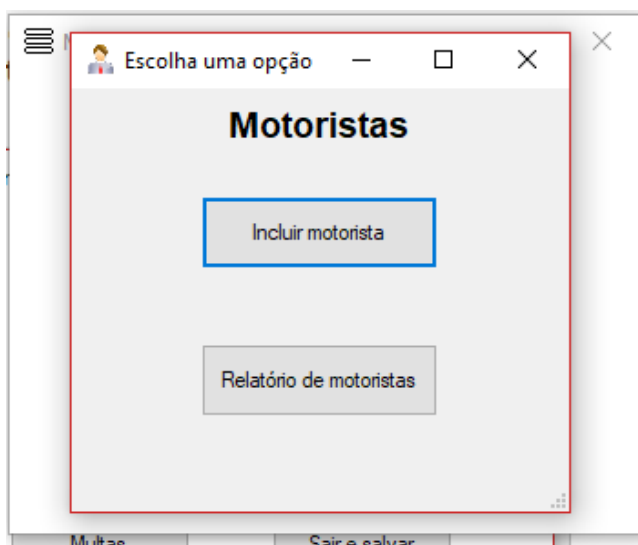
Começamos o desenvolvimento abordando sobre motoristas e a carteira nacional de habilitação, sobre os tipos de carteira, registro e emplacamento dos veículos, multas e suas pontuações, logo após mostramos o diagrama UML a explicação e os códigos do programa, concluimos e apresentamos as referências.

1.3 Utilização do Programa

Ao abrir o programa nos deparamos com menu com as seguintes opções: **Motoristas, Veiculos, Multas e Sair e Salvar.**



Ao clicar em Motoristas se tem as seguintes opções:



Em incluir motorista temos os campos para inserir os dados do novo motorista, o botão para incluir e uma tabela com os motoristas existentes e seus dados.

Incluir novo motorista

Insira os dados para incluir um novo motorista

Nome:

CPF:

Data de nascimento: , de de

	Nome	CPF	Número CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111.111.111-11	06540403721	12	26/06/1998	Ativa
	Igor Gabriel	222.222.222-22	31506423546	3	26/06/1998	Ativa
	Igor Palhares	333.333.333-33	37044121240	7	26/06/1998	Ativa
	Guilherme	444.444.444-44	51360416108	5	26/06/1998	Ativa
	João Vitor	555.555.555-55	81720108153	7	26/06/1998	Ativa
	Gabriel Silva	666.666.666-66	58242038085	0	26/06/1998	Ativa
	Maria Silva	777.777.777-77	85083881815	0	26/06/1998	Ativa
	Pedro Henrique	888.888.888-88	23524024807	0	26/06/1998	Ativa
*						

Em relatórios temos uma lista com motoristas e opções para a filtragem de determinado motorista.

Relatório de motoristas

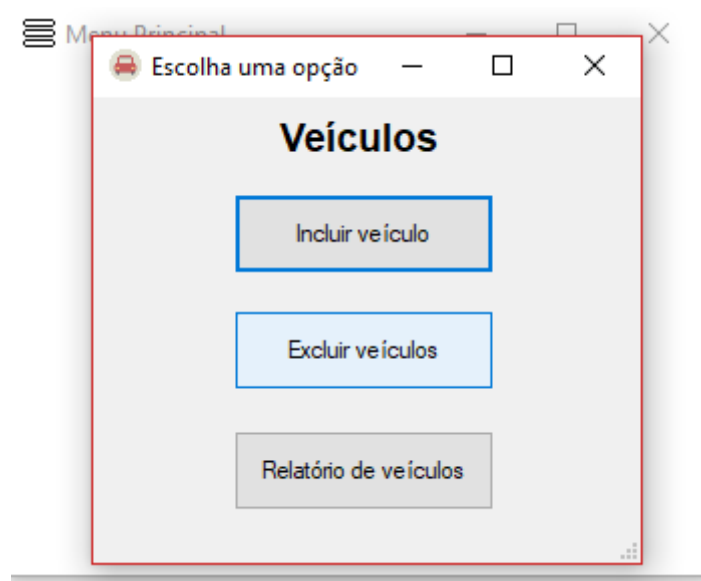
Escolha o filtro para o relatório de motoristas

☒ Todos ☐ Possuem carteira ativa ☐ Não possuem multas

Pesquisar motorista por CPF:

	Nome	CPF	Número da CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111.111.111-11	06540403721	12	26/06/1998	Ativa
	Igor Gabriel	222.222.222-22	31506423546	3	26/06/1998	Ativa
	Igor Palhares	333.333.333-33	37044121240	7	26/06/1998	Ativa
	Guilherme	444.444.444-44	51360416108	5	26/06/1998	Ativa
	João Vitor	555.555.555-55	81720108153	7	26/06/1998	Ativa
	Gabriel Silva	666.666.666-66	58242038085	0	26/06/1998	Ativa
	Maria Silva	777.777.777-77	85083881815	0	26/06/1998	Ativa
	Pedro Henrique	888.888.888-88	23524024807	0	26/06/1998	Ativa
*						

Em veículos temos as opções de inserir, excluir e gerar relatório.



Em inserir veículo temos os campos para a inserção de dados do veículo, o botão para inserir e uma tabela com os motoristas cadastrados.

The screenshot shows a window titled 'Incluir novo veículo' (Include new vehicle) with a car icon. The heading 'Insira os dados para incluir um novo veículo' (Enter the data to include a new vehicle) is centered. Below it are input fields for 'Placa' (Plate), 'Chassi' (Chassis), 'Modelo' (Model), and 'Ano' (Year), along with a 'Gerar placa' (Generate plate) button and a 'CPF do proprietário' (Owner's CPF) field. A 'Cadastrar veículo' (Register vehicle) button is at the bottom. Below the button is a table titled 'Motoristas cadastrados' (Registered drivers).

	Nome	CPF	Número da CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111.111.111-11	06540403721	12	26/06/1998	Possui
	Igor Gabriel	222.222.222-22	31506423546	3	26/06/1998	Possui
	Igor Palhares	333.333.333-33	37044121240	7	26/06/1998	Possui
	Guilherme	444.444.444-44	51360416108	5	26/06/1998	Possui
	João Vitor	555.555.555-55	81720108153	7	26/06/1998	Possui
	Gabriel Silva	666.666.666-66	58242038085	0	26/06/1998	Possui
	Maria Silva	777.777.777-77	85083881815	0	26/06/1998	Possui
	Pedro Henrique	888.888.888-88	23524024807	0	26/06/1998	Possui
*						

Em Excluir Veiculo temos o campo para a inserção da placa do veículo a ser excluído, o botão para excluir e logo abaixo temos uma tabela com os veículos existentes.

Excluir um veículo

Digite a placa do veículo a ser excluído:

Excluir veículo

	Placa	Chassi	Modelo	Ano	Nome proprietário	CPF Proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha	111.111.111-11
	ESR-4221	34567	Gol	2014	Paulo Rocha	111.111.111-11
	GLB-6474	432566	Jeep	2016	Igor Gabriel	222.222.222-22
	WGM-5722	423432	Jeep	2019	Igor Palhares	333.333.333-33
	FLB-4260	3245263	Palio	2012	Guilherme	444.444.444-44
	CYQ-4504	3452687	Fox	2016	Gabriel Silva	666.666.666-66
	VRQ-5874	23563	Uno	2017	Maria Silva	777.777.777-77
*						

Em relatório se tem campos para a filtragem de veículo e uma tabela com os veículos existentes.

Relatório de veículos

Escolha o filtro para o relatório de veículos

☒ Todos

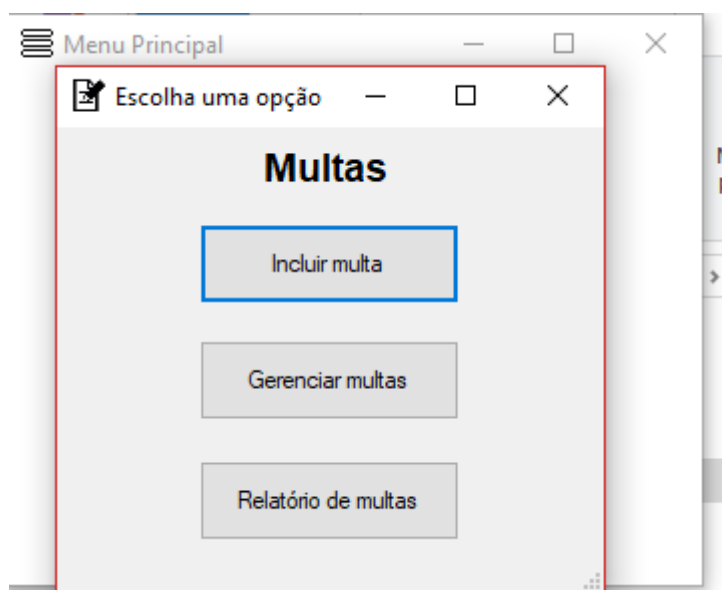
Pesquisar por placa:

Pesquisar por CPF:

Filtrar

	Placa	Chassi	Modelo	Ano	Nome proprietário	CPF Proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha	111.111.111-11
	ESR-4221	34567	Gol	2014	Paulo Rocha	111.111.111-11
	GLB-6474	432566	Jeep	2016	Igor Gabriel	222.222.222-22
	WGM-5722	423432	Jeep	2019	Igor Palhares	333.333.333-33
	FLB-4260	3245263	Palio	2012	Guilherme	444.444.444-44
	CYQ-4504	3452687	Fox	2016	Gabriel Silva	666.666.666-66
	VRQ-5874	23563	Uno	2017	Maria Silva	777.777.777-77
*						

Em Multas temos podemos inserir, gerenciar e gerar os relatórios das multas.



Em Incluir Multa temos os campos para inserção de dados necessários para gerar uma multa, o botão para aplicar e logo abaixo informações sobre veículos e motoristas.

The image shows a software window titled 'Incluir uma nova multa' with a title bar containing a document icon, a minus sign, a maximize button, and a close button. The main content area is titled 'Insira os dados para incluir uma nova multa' and contains the following fields and controls:

- Digite a placa:
- Motivo da multa:
- Gravidade:
- CPF do motorista:
- Data:
- Valor da multa:
- Aplicar multa:

Below the form is a table with 6 columns: Placa, Chassi, Modelo, Ano, and Nome proprietário. The first row is highlighted in blue.

	Placa	Chassi	Modelo	Ano	Nome proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha
	ESR-4221	34567	Gol	2014	Paulo Rocha
	GLB-6474	432566	Jeep	2016	Igor Gabriel
	WGM-5722	423432	Jeep	2019	Igor Palhares
	FLB-4260	3245263	Palio	2012	Guilherme
	CYQ-4504	3452687	Fox	2016	Gabriel Silva
	VRQ-5874	23563	Uno	2017	Maria Silva
*					

At the bottom of the table is a scroll bar.

Em gerenciar multa podemos ter uma tabela com veículos e motoristas, e um campo para inserir a placa do veículo e verificar suas multas.

Gerenciar multas

Veículos

Placa para verificar multas:

	Placa	Chassi	Modelo	Ano	Nome proprietário
	NGC-4271	532532	Uno	2012	Paulo Rocha
	ESR-4221	34567	Gol	2014	Paulo Rocha
▶	GLB-6474	432566	Jeep	2016	Igor Gabriel
	WGM-5722	423432	Jeep	2019	Igor Palhares
	FLB-4260	3245263	Palo	2012	Guilherme
	CYQ-4504	3452687	Fox	2016	Gabriel Silva
	VRQ-5874	23563	Uno	2017	Maria Silva
*					

Multas

☒ Quitar uma multa
 ☐ Transferir uma multa
 ☐ Justificar uma multa

ID da multa:

Placa:

CPF do Multado:

	ID da multa	Placa	Motivo	Paga	Gravidade
*					

Inserindo ou clicando na placa podemos ver detalhes das multas, quitar, transferir ou justificar uma multa.

Gerenciar multas

Veículos

Placa para verificar multas:

	Placa	Chassi	Modelo	Ano	Nome proprietário
	NGC-4271	532532	Uno	2012	Paulo Rocha
	ESR-4221	34567	Gol	2014	Paulo Rocha
▶	GLB-6474	432566	Jeep	2016	Igor Gabriel
	WGM-5722	423432	Jeep	2019	Igor Palhares
	FLB-4260	3245263	Palo	2012	Guilherme
	CYQ-4504	3452687	Fox	2016	Gabriel Silva
	VRQ-5874	23563	Uno	2017	Maria Silva
*					

Multas

☒ Quitar uma multa
 ☐ Transferir uma multa
 ☐ Justificar uma multa

ID da multa:

Placa:

CPF do Multado:

	ID da multa	Placa	Motivo	Paga	Gravidade
▶	1	GLB-6474	Estacionar em lo...	Não	Leve
	2	GLB-6474	Ultrapassar faixa ...	Não	Gravíssima
*					

Em relatório de multas é possível ver todas as multas existentes e filtrar por multas pagas, multas não pagas, multas transferidas, multas justificadas, por placa e CPF.

Relatório de multas

Escolha o filtro para o relatório de multas

☒ Todas
☐ Multas pagas
☐ Multas não pagas
☐ Multas transferidas
☐ Multas justificadas

Filtrar multas por placa:

Filtrar multas por CPF:

	Placa	Motivo	Paga	Gravidade	Data da multa	Valor	Penalidade	CPF do multado	Transferida	Justificada
▶	NGC-4271	Ultrapassar sinal ...	Não	Grave	06/06/2019	256	5	111.111.111-11	Não	Não
	GLB-6474	Estacionar em lo...	Não	Leve	06/06/2019	103	3	222.222.222-22	Não	Não
	GLB-6474	Ultrapassar faixa ...	Não	Gravíssima	07/06/2019	321	7	333.333.333-33	Não	Não
	WGM-5722	Ultrapassagem n...	Não	Gravíssima	06/06/2019	352	7	111.111.111-11	Não	Não
	CYQ-4504	Dirigir com farol d...	Não	Grave	06/06/2019	167	5	444.444.444-44	Não	Não
	VRQ-5874	Direção perigosa	Não	Gravíssima	06/06/2019	309	7	555.555.555-55	Não	Não
*										

Sair e Salvar, salva as modificações e encerra o programa.

2. Desenvolvimento

2.1 Motoristas

A carteira nacional de habilitação (CNH), certifica que o cidadão Brasileiro pode dirigir em todo território nacional. A pessoa que quiser tirar a primeira CNH precisa seguir alguns passos: Ao completar 18 anos, qualquer cidadão brasileiro que saiba ler e escrever e possua os documentos de identidade RG e CPF está habilitado a tirar a primeira CNH. O primeiro passo é ir em algum dos Departamentos de Trânsito de seu estado em uma Autoescola para fazer seu cadastro onde serão coletadas as digitais de cada dedo e tiradas as fotos do futuro motorista. O próximo passo, o Detran vai indicar ao cidadão as clínicas credenciadas pela repartição para realizar os exames de aptidão física e psicológica, obrigatórios para tirar a primeira habilitação. A pessoa deve comparecer à clínica de preferência (dentre as autorizadas pelo Detran) e realizar o pagamento das taxas dos exames diretamente a ela. Após a aprovação no exame, a pessoa pode procurar uma Autoescola para começar os estudos na legislação que são 45 horas de aula em sala para o aluno aprender sobre as regras de trânsito. Vale lembrar que esta etapa é a mesma para quem está tirando a carteira de moto (A) e de carro (B). Ao final do curso, você é obrigado a realizar uma prova de múltipla escolha, a prova terá 30 questões e a pessoa terá que tirar acima de 70%, você estará habilitado a seguir para as aulas práticas.

O próximo passo é fazer até 5 horas de aulas no simulador, incluindo uma hora que seria reservada a uma aula prática noturna. Após o termino das aulas do simulador, o aluno irá começar as aulas nas ruas, lembrando que são exigidas no mínimo 20 horas de aulas práticas, enquanto que para automóveis são 25 horas pelo menos, sendo 5 horas em aulas noturnas (este total inclui as 5 horas de simulador).

A quantidade de aulas práticas varia de acordo com o tipo de carteira. No caso das motos, são exigidas no mínimo 20 horas de aulas práticas, enquanto que

para automóveis são 25 horas pelo menos, sendo 5 horas em aulas noturnas (este total inclui as 5 horas de simulador).

Terminada a carga horária do curso prático, é hora de realizar a prova prática. O exame nada mais é do que a reprodução exata daquilo que você treinou durante as mais de 20 horas na autoescola. Para a carteira B, o trajeto envolve parada em ruas perpendiculares, acíves e baliza. Já os futuros motociclistas têm que realizar um circuito praticamente idêntico ao que vieram treinando anteriormente. Após aprovação neste exame, o aluno receberá a PPD – permissão para dirigir, que tem validade de 1 ano. Excedendo este tempo o motorista poderá pecar a CNH definitiva.

2.1.1. Tipos de carteira

O Código de Trânsito Brasileiro divide a habilitação para dirigir em cinco categorias:

A: condutor de veículo motorizado de duas ou três rodas, com ou sem carro lateral (motos);

B: condutor de veículo motorizado não abrangido pela categoria com peso bruto total inferior a 3.500 quilos e lotação máxima de oito lugares, além do motorista (automóveis);

C: condutor de veículo motorizado usado para transporte de carga, com peso bruto superior a 3.500 quilos (como caminhões);

D: condutor de veículo motorizado usado no transporte de passageiros, com lotação superior a oito lugares além do motorista (ônibus e vans, por exemplo);

E: condutor de combinação de veículos em que a unidade conduzida se enquadre nas categorias B, C ou D e cuja unidade acoplada ou rebocada tenha peso bruto de 6 mil quilos ou mais; ou cuja lotação seja superior a oito lugares; ou, ainda, que seja enquadrado na categoria trailer.

A primeira Carteira Nacional de Habilitação só pode ser retirada nas categorias, A, B ou AB. No último caso, a pessoa deve participar de dois cursos preparatórios. Quem possui habilitação na categoria B pode mudá-la para C ou D; quem possui na categoria C, pode mudar para as categorias D ou E; e quem possui o documento na categoria D, pode obtê-lo na categoria E. Para solicitar a admissão nas categorias C, D ou E, o condutor deve estar habilitado há pelo menos um ano na categoria anterior exigida (por exemplo, para obter a CNH na categoria C, é necessário ter no mínimo um ano de habilitação na categoria B).

2.2 Veículos

2.2.1. Registrando um novo carro

Todo veículo quando fabricado possui um cartão de identificação que possui chassi, espécie, combustível, cor e as peças utilizadas em sua montagem. Quando a montagem é terminada esses dados são enviados para a BIN, que é uma base de dados informatizada e centralizada que armazena informações oficiais do DENATRAN (Departamento Nacional de Trânsito), contendo características e informações dos veículos pertencentes à frota nacional a partir do sistema de Registro Nacional de Veículo (RENAVAM), esse processo é chamado de pré-cadastro.

2.2.2. Emplacando carro novo

Deve-se registrar o veículo 0 km em até 30 dias após a emissão da nota fiscal para gerar o Certificado de Registro do Veículo (CRV). Esse documento permite o emplacamento e a concessão do Certificado de Registro e Licenciamento do Veículo (CRLV), principal documento de porte obrigatório do carro.

Compareça ao Detran para emitir os certificados e obter o número da placa, assim como os valores do IPVA e DPVAT. Os preços para o registro do veículo e emplacamento variam de cidade para cidade e a placa pode ser personalizada por R\$100,00.

Os documentos necessários são: original e cópia do RG, CPF e CNH, cópia de um comprovante de residência recente, nota fiscal original da concessionária com decalque do chassi, duas cópias do formulário do RENAVAM preenchido e, caso o carro seja financiado, é necessário também levar o contrato original e uma cópia assinada por todos os envolvidos.

Lá é necessário pagar as taxas de emplacamento, escolher a placa e dar entrada ao CRV. Assim que o certificado ficar disponível é necessário voltar ao Detran.

2.3 Multas

No Brasil, as infrações de trânsito são penalizadas em multa e pontuação na carteira ou permissão para dirigir, o limite de pontos permitidos na CNH é 19 pontos, após isso a carteira é suspensa, esses pontos se acumulam no período de um ano, após

12 meses a pontuação da multa expira.

As multas se encaixam em quatro categorias:

Leve: 3 pontos – R\$ 88,38

Exemplos: estacionar o veículo nos acostamentos (art. 181, VII); parar o veículo na faixa de pedestres (art. 182, VI); usar buzina em desacordo com as normas estabelecidas pelo CONTRAN (art. 227, V).

Leve: 4 pontos – R\$ 130,16.

Exemplo: atirar do veículo ou abandonar na via objetos ou substâncias (art. 172); estacionar o veículo na contração da direção (art. 182, XV); não mudar de pista com antecedência para dobrar (art. 197).

Grave: 5 pontos – R\$ 195,23.

Exemplo: estacionar o veículo em fila dupla (art. 181, XI); deixar de dar preferência a pedestre quando houver iniciado a travessia (art. 214, IV); conduzir pessoas, animais ou carga na parte externa do veículo (art. 235).

Gravíssima: 7 pontos – R\$ 295,47.

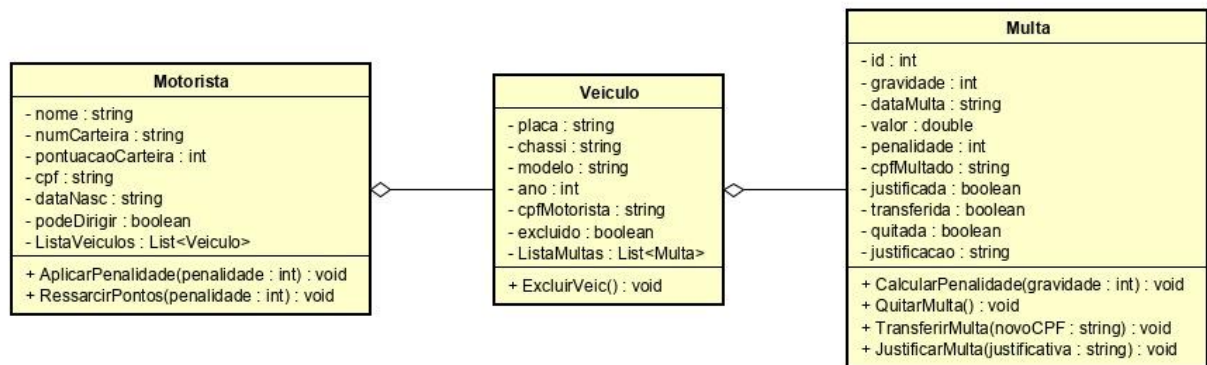
O preço da multa gravíssima contém o chamado fator multiplicador, podendo ser multiplicado por 3,5,10 ou até 60x.

Exemplo: fazer falsa declaração de domicílio para fins de registro, licenciamento ou habilitação (art. 242); bloquear a via com veículo (art. 253); dirigir veículo com a CNH cassada ou suspensa (art. 162, II).

Os pontos são atribuídos ao condutor da infração, quando ela é registrada por radares, a multa é atribuída ao proprietário do veículo.

2.4 Códigos e UML

2.4.2 UML



2.4.3 Códigos e Interfaces Gráficas

2.4.3.1 Códigos da classe Controle

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;
```

```
namespace TFI
```

```
{
```

```
    class Controle
```

```
    {
```

```
        public static List<Motorista> ListaMotoristas = new List<Motorista>();
```

```
        public static void VerificarArquivos() { //Verifica se os arquivos existem, caso
        falta algum ele cria antes do form iniciar
```

```
            if (!File.Exists("motoristas.txt") || !File.Exists("veiculos.txt")
            || !File.Exists("multas.txt")) { //Caso algum arquivo não tenha sido criado, ele cria.
```

```
                if (!File.Exists("motoristas.txt")) {
                    FileStream fs = File.Create("motoristas.txt");
```

```

        fs.Close();
    }
    if (!File.Exists("veiculos.txt")) {
        FileStream fs = File.Create("veiculos.txt");
        fs.Close();
    }
    if (!File.Exists("multas.txt")) {
        FileStream fs = File.Create("multas.txt");
        fs.Close();
    }
    MessageBox.Show("Um ou mais arquivos estão faltando.\nOs arquivos
que estão faltando foram criados.", "Aviso", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
}
}

```

```

public static void leArqMultas(Veiculo v) {

```

```

    StreamReader lerMultas = new StreamReader("multas.txt"); //abrir o arquivo.

```

```

    string linhaMultas;

```

```

    string[] auxSeparador; //separar dados de cada linha por strings

```

```

    int gravidade, penalidade;

```

```

    string dataMulta, placa, cpfM, justificacao;

```

```

    double valor;

```

```

    bool transferida, quitada, justificada;

```

```

    Multa m;

```

```

    linhaMultas = lerMultas.ReadLine(); //faz a primeira leitura (1a linha)

```

```

    while (linhaMultas != null) { //enquanto houver informações..

```

```

        auxSeparador = linhaMultas.Split(';'); //separa a string linha em um vetor
de varias strings (separadas pelo ';')

```

```

        placa = auxSeparador[0];

```

```

        if (placa == v.Placa) {

```

```

            //id = int.Parse(auxSeparador[1]);

```

```

            gravidade = int.Parse(auxSeparador[2]);

```

```

            dataMulta = auxSeparador[3];

```

```

            valor = double.Parse(auxSeparador[4]);

```

```

            penalidade = int.Parse(auxSeparador[5]);

```

```

            cpfM = auxSeparador[6];

```

```

            transferida = bool.Parse(auxSeparador[7]);

```

```

            quitada = bool.Parse(auxSeparador[8]);

```

```

            justificada = bool.Parse(auxSeparador[9]);

```

```

        m = new Multa(v.ListaMultas1.Count, gravidade, dataMulta, valor,
penalidade, cpfM, transferida, quitada, justificada); //Cria a multa

        if (m.Justificada == true) { //Caso a multa seja justificada
            justificacao = auxSeparador[10];
            m.Justificacao = justificacao; //Inserir a justificação
        }

        v.ListaMultas1.Add(m); //Insere a multa na lista de multas do veiculo
    }

    linhaMultas = lerMultas.ReadLine();
}
lerMultas.Close(); // fecha o arquivo
}

public static void leArqVeiculos(Motorista mot) {

    StreamReader lerVeiculo = new StreamReader("veiculos.txt"); //abrir o
arquivo

    string linhaVeiculo;
    string[] auxSeparador; //separar dados de cada linha por strings

    //dados do veiculo
    string placa, modelo, cpfMot, chassi;
    int ano;
    bool excluido;
    Veiculo v;

    linhaVeiculo = lerVeiculo.ReadLine(); //faz a primeira leitura (1a linha)

    while (linhaVeiculo != null) { //enquanto houver informações..

        auxSeparador = linhaVeiculo.Split(';'); //separa a string linha em um vetor
de varias strings (separadas pelo ';')

        cpfMot = auxSeparador[4];

        if (cpfMot == mot.Cpf) {

            placa = auxSeparador[0];
            chassi = auxSeparador[1];
            modelo = auxSeparador[2];
            ano = int.Parse(auxSeparador[3]);
            excluido = bool.Parse(auxSeparador[5]);

```

```

        v = new Veiculo(placa, chassi, modelo, ano, cpfMot, excluido); //Cria o
veiculo

        mot.ListaVeiculos1.Add(v); //Adiciona o veiculo na lista de veiculos do
motorista

        leArqMultas(v); //Le o arquivo de multas para conferir se existe alguma
multa relacionada nesse veiculo
    }

    linhaVeiculo = lerVeiculo.ReadLine();
}
lerVeiculo.Close(); // fecha o arquivo
}

public static void leArquivos() {

    StreamReader lerMotorista = new StreamReader("motoristas.txt"); //abrir o
arquivo

    string linhaMot;    //ler e escrever as linhas do arquivo
    string[] auxSeparador; //separar dados de cada linha por strings

    Motorista mot;

    //dados do motorista
    int pontuacaoCarteira;
    string nome, cpf, dataNasc, numCarteira;
    bool podeDirigir;

    linhaMot = lerMotorista.ReadLine(); //faz a primeira leitura (1a linha)

    while (linhaMot != null) {    //enquanto houver informações..

        auxSeparador = linhaMot.Split(';'); //separa a string linha em um vetor de
varias strings (separadas pelo ';')

        nome = auxSeparador[0];
        numCarteira = auxSeparador[1];
        pontuacaoCarteira = int.Parse(auxSeparador[2]);
        cpf = auxSeparador[3];
        dataNasc = auxSeparador[4];
        podeDirigir = bool.Parse(auxSeparador[5]);

        mot = new Motorista(nome, numCarteira, pontuacaoCarteira, cpf,
dataNasc, podeDirigir); // Criando um motorista

```

```
Controle.ListaMotoristas.Add(mot); // Adicionando o motorista na lista de motoristas
```

```
leArqVeiculos(mot); // Lê o arquivo de motoristas para verificar se esse motorista possui um veículo ou não
```

```
    linhaMot = lerMotorista.ReadLine();  
}
```

```
lerMotorista.Close(); // fecha o arquivo  
}
```

```
public static void gravarArquivos() {  
    StreamWriter escreverMot = new StreamWriter("motoristas.txt"); //abrir o arquivo  
    StreamWriter escreverVeic = new StreamWriter("veiculos.txt"); //abrir o arquivo  
    StreamWriter escreverMultas = new StreamWriter("multas.txt"); //abrir o arquivo
```

```
    string linha;
```

```
    foreach (Motorista mot in Controle.ListaMotoristas) { //Percorre a lista de motoristas  
        linha = mot.Nome + ";" + mot.NumCarteira + ";" + mot.PontuacaoCarteira +  
        ";" + mot.Cpf + ";" + mot.DataNasc + ";" + mot.PodeDirigir;  
        escreverMot.WriteLine(linha); //Escreve as informações dos motoristas no arquivo
```

```
        foreach (Veiculo v in mot.ListaVeiculos1) { //Percorre a lista de veículos  
            linha = v.Placa + ";" + v.Chassi + ";" + v.Modelo + ";" + v.Ano + ";" +  
            v.CpfMotorista + ";" + v.Excluido;  
            escreverVeic.WriteLine(linha); //Escreve as informações dos veículos no arquivo
```

```
            foreach (Multa m in v.ListaMultas1) { //Percorre a lista de multas  
                if (m.Justificada == false) { //Se a multa não for justificada insere normalmente  
                    linha = v.Placa + ";" + m.Id + ";" + m.Gravidade + ";" + m.DataMulta  
                    + ";" + m.Valor + ";" + m.Penalidade + ";" + m.CpfMultado + ";" + m.Transferida + ";" +  
                    m.Quitada + ";" + m.Justificada;  
                } else { //Caso a multa seja justificada, insere junto com a justificação da mesma  
                    linha = v.Placa + ";" + m.Id + ";" + m.Gravidade + ";" + m.DataMulta  
                    + ";" + m.Valor + ";" + m.Penalidade + ";" + m.CpfMultado + ";" + m.Transferida + ";" +  
                    m.Quitada + ";" + m.Justificada + ";" + m.Justificacao;  
                }  
            }  
        }  
    }  
}
```

```

        escreverMultas.WriteLine(linha); //Escreve as informações das multas
no arquivo
    }
}

    escreverMot.Close(); //fecha o arquivo
    escreverVeic.Close(); //fecha o arquivo
    escreverMultas.Close(); //fecha o arquivo
}
}
}

```

2.4.3.2 Explicação dos Códigos da classe Controle

A classe controle é o coração do sistema, é onde é criado o vetor de motoristas, o método para verificar se os arquivos existem ou não (caso não ele cria) e métodos para leitura (ler e salvar no vetor) e escrita dos arquivos para salvar os dados.

2.4.3.3 Códigos da classe Motoristas

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TFI {
    class Motorista {
        //Atributos
        private string nome;
        private string numCarteira;
        private int pontuacaoCarteira;
        private string cpf;
        private string dataNasc;
        private bool podeDirigir;
        private List<Veiculo> ListaVeiculos = new List<Veiculo>();

        public Motorista(string nome, string num, int pontuacao, string cpf, string dataN,
bool podeDirigir) { //Construtor
            Nome = nome;
            NumCarteira = num;
            PontuacaoCarteira = pontuacao;
            Cpf = cpf;
            DataNasc = dataN;

```



```

    PodeDirigir = podeDirigir;
}

//Métodos de instancia

public void AplicarPenalidade(int penalidade) {
    PontuacaoCarteira += penalidade;

    if (PontuacaoCarteira >= 20) {
        PodeDirigir = false;
    } else {
        PodeDirigir = true;
    }
}

public void RessarcirPontos(int penalidade) {
    PontuacaoCarteira -= penalidade;

    if (PontuacaoCarteira >= 20) {
        PodeDirigir = false;
    } else {
        PodeDirigir = true;
    }
}

//Getters e setters

public string Nome {
    get {
        return nome;
    }

    set {
        nome = value;
    }
}

public string NumCarteira {
    get {
        return numCarteira;
    }

    set {
        numCarteira = value;
    }
}

public int PontuacaoCarteira {

```

```

    get {
        return pontuacaoCarteira;
    }

    set {
        pontuacaoCarteira = value;
    }
}

public string Cpf {
    get {
        return cpf;
    }

    set {
        cpf = value;
    }
}

public string DataNasc {
    get {
        return dataNasc;
    }

    set {
        dataNasc = value;
    }
}

public bool PodeDirigir {
    get {
        return podeDirigir;
    }

    set {
        podeDirigir = value;
    }
}

internal List<Veiculo> ListaVeiculos1 {
    get {
        return ListaVeiculos;
    }

    set {
        ListaVeiculos = value;
    }
}
}
}

```

2.4.3.4 Explicação dos Códigos da classe Motorista

A classe Motorista contém atributos relacionados a um motorista (nome, cpf, pontuação da carteira, data de nascimento e etc) um dos atributos é uma lista de veículos, pois o motorista pode ou não ter um veículo (ou pode possuir vários veículos). Possui também um método para aplicar a penalidade de uma multa (colocar os pontos na carteira de acordo com a penalidade de uma multa recebida) e para ressarcir pontos (em caso de justificar multas por exemplo).

2.4.3.5 Códigos da classe Veículo

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TFI {
    class Veiculo {
        //Atributos
        private string placa;
        private string chassi;
        private string modelo;
        private int ano;
        private string cpfMotorista;
        private bool excluido;
        private List<Multa> ListaMultas = new List<Multa>();

        public Veiculo(string placa, string chassi ,string modelo, int ano, string cpfMot,
bool excluido) { //Construtor
            Placa = placa;
            Modelo = modelo;
            Ano = ano;
            CpfMotorista = cpfMot;
            Chassi = chassi;
            Excluido = excluido;
        }

        //Métodos de instancia
        public void ExcluirVeic() {
            Excluido = true;
        }

        //Getters e setters

        public string Placa {
            get {
```

```

        return placa;
    }

    set {
        placa = value;
    }
}

public string Chassi {
    get {
        return chassi;
    }

    set {
        chassi = value;
    }
}

public string Modelo {
    get {
        return modelo;
    }

    set {
        modelo = value;
    }
}

public int Ano {
    get {
        return ano;
    }

    set {
        ano = value;
    }
}

public string CpfMotorista {
    get {
        return cpfMotorista;
    }

    set {
        cpfMotorista = value;
    }
}

```

```

    public bool Excluido {
        get {
            return excluido;
        }

        set {
            excluido = value;
        }
    }

    internal List<Multa> ListaMultas1 {
        get {
            return ListaMultas;
        }

        set {
            ListaMultas = value;
        }
    }
}

```

2.4.3.6 Explicação dos Códigos da classe Veículo

A classe veículos possui atributos de um veículo, como placa, chassi, modelo, CPF do motorista proprietário e etc, possui também uma lista de multas como atributo, pois num veículo pode estar registrado várias multas ou nenhuma. Contém um método de instância para excluir o veículo (deixa-lo “desativado” por assim dizer, de forma que o registro do veículo ainda permaneça.

2.4.3.6 Códigos da classe Multa

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace TFI {
    class Multa {
        //Atributos
        private int id;
        private int gravidade; // 1,2,3,4
        private string dataMulta;
        private double valor;
    }
}

```

```
private int penalidade;  
private string cpfMultado;  
private bool justificada;  
private bool transferida;  
private bool quitada;  
private string justificacao;
```

```
public Multa(int id, int gravidade, string dataMulta, double valor, int penalidade,  
string cpfM, bool transferida, bool quitada, bool justificada) { //Construtor para a  
leitura (apenas carrega dados)
```

```
    Id = id;  
    Gravidade = gravidade;  
    DataMulta = dataMulta;  
    Valor = valor;  
    Penalidade = penalidade;  
    CpfMultado = cpfM;  
    Transferida = transferida;  
    Quitada = quitada;  
    Justificada = justificada;  
}
```

```
public Multa(int id, int gravidade, string dataMulta, double valor, string cpfM,  
bool transferida, bool quitada, bool justificada) { //Construtor para a aplicação da  
multa (apenas carrega dados)
```

```
    Id = id;  
    Gravidade = gravidade;  
    DataMulta = dataMulta;  
    Valor = valor;  
    CpfMultado = cpfM;  
    Transferida = transferida;  
    Quitada = quitada;  
    Justificada = justificada;  
}
```

```
//Métodos de instância
```

```
public void CalcularPenalidade(int gravidade) {  
    if (gravidade == 0) Penalidade = 3; // leve  
    else if (gravidade == 1) Penalidade = 4; // media  
    else if (gravidade == 2) Penalidade = 5; // grave  
    else if (gravidade == 3) Penalidade = 7; // gravissima  
    else Penalidade = 0;  
}
```

```
public void QuitarMulta() {  
    Quitada = true;  
}
```

```
public void TransferirMulta(string novoCPF) {  
    CpfMultado = novoCPF;  
    Transferida = true;  
}
```

```
public void JustificarMulta(string justificativa) {  
    Justificacao = justificativa;  
    Justificada = true;  
}
```

//Getters e setters

```
public int Id {  
    get {  
        return id;  
    }  
  
    set {  
        id = value;  
    }  
}
```

```
public int Gravidade {  
    get {  
        return gravidade;  
    }  
  
    set {  
        gravidade = value;  
    }  
}
```

```
public string DataMulta {  
    get {  
        return dataMulta;  
    }  
  
    set {  
        dataMulta = value;  
    }  
}
```

```
public double Valor {  
    get {  
        return valor;  
    }  
  
    set {
```

```

        valor = value;
    }
}

public int Penalidade {
    get {
        return penalidade;
    }

    set {
        penalidade = value;
    }
}

public string CpfMultado {
    get {
        return cpfMultado;
    }

    set {
        cpfMultado = value;
    }
}

public bool Justificada {
    get {
        return justificada;
    }

    set {
        justificada = value;
    }
}

public bool Transferida {
    get {
        return transferida;
    }

    set {
        transferida = value;
    }
}

public bool Quitada {
    get {
        return quitada;
    }
}

```



```

        set {
            quitada = value;
        }
    }

    public string Justificacao {
        get {
            return justificacao;
        }

        set {
            justificacao = value;
        }
    }
}

```

2.4.3.7 Explicação dos Códigos da classe Multas

A classe multas possui atributos relacionados a uma multa como por exemplo: gravidade, data, valor, penalidade (em pontos) entre outras, além disso possui métodos de instância para calcular a penalidade de uma multa (a partir da gravidade), quitar uma multa, transferir e justificar.

2.4.3.8 Interfaces gráficas e seus códigos, começando pelo Menu Principal (frmPrincipal)



2.4.3.8.1 Código do frmPrincipal:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.IO;

/*
Trabalho interdisciplinar de programação orientada a objetos (Prática e Teórica).
Sistemas de Informação, Manhã - Segundo Período
Professor: Paulo Amaral
Projeto: TFI
Data: 02/06/2019
Nome dos alunos: Paulo Henrique, Igor Palhares
Descricao: Este trabalho é um sistema de motoristas, veiculos e multas, para multar
veiculos/motoristas, alterar multas (justificar, pagar, transferir), administrar os pontos
na carteira de motoristas e seus veiculos, apresenta tambem uma parte somente
para relatórios, para obter relatório mais especificos (usando filtros) ou mais gerais.

*/

namespace TFI {
    public partial class frmPrincipal : Form {
        public frmPrincipal() {
            InitializeComponent();
            Controle.VerificarArquivos(); //Verifica os arquivos para que realize a leitura
            corretamente
            Controle.leArquivos(); //Le os arquivos na propria inicialização do sistema
        }

        private void frmPrincipal_Load(object sender, EventArgs e) {

        }

        private void btnSair_Click(object sender, EventArgs e) {
            Controle.gravarArquivos(); //Grava as informações nos arquivos antes de sair
            this.Close(); //Fecha o programa
        }

        private void frmPrincipal_FormClosing(Object sender, FormClosingEventArgs e)
        { //Evento formClosing para gravar no arquivo
            Controle.gravarArquivos();
        }
    }
}
```

```

    }

    private void btnMotoristas_Click(object sender, EventArgs e) {
        frmOpcoesMotorista opMotorista = new frmOpcoesMotorista();
        opMotorista.ShowDialog();
    }

    private void btnVeiculos_Click(object sender, EventArgs e) {
        frmOpcoesVeiculo opVeiculo = new frmOpcoesVeiculo();
        opVeiculo.ShowDialog();
    }

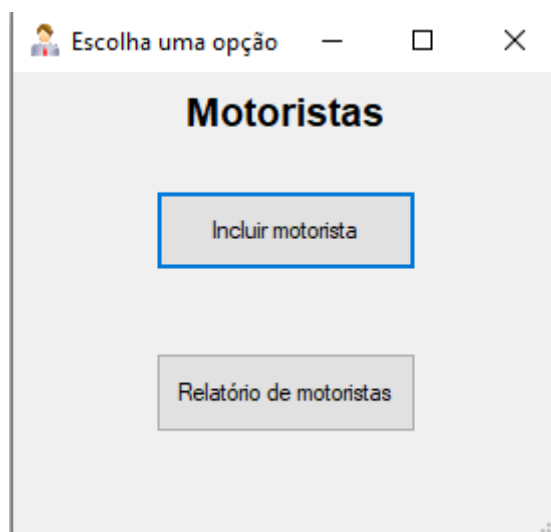
    private void btnMultas_Click(object sender, EventArgs e) {
        frmOpcoesMulta opMulta = new frmOpcoesMulta();
        opMulta.ShowDialog();
    }
}
}

```

2.4.3.8.2 Explicação Código frmPrincipal:

O form principal é o form de menu, onde possui todas as opções e exibe as funcionalidades do programa logo de cara, de forma a facilitar para o usuário, sem que fique com funções e telas “escondidas”. Esse form tem como simples tarefa conter botões que abrem outros forms de acordo com suas funcionalidades.

2.4.3.9 Opções para Motorista (frmOpcoesMotorista)



2.4.3.9.1 Código do frmOpcoesMotorista:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmOpcoesMotorista : Form {
        public frmOpcoesMotorista() {
            InitializeComponent();
        }

        private void btnIncluirMotorista_Click(object sender, EventArgs e) {
            frmIncluirMotorista incluirMotorista = new frmIncluirMotorista();
            incluirMotorista.ShowDialog();
        }

        private void btnRelatorioMotoristas_Click(object sender, EventArgs e) {
            frmRelatorioMotoristas relatMot = new frmRelatorioMotoristas();
            relatMot.ShowDialog();
        }

        private void frmOpcoesMotorista_Load(object sender, EventArgs e)
        {
        }
    }
}
```

2.4.3.9.2 Explicação Código do frmOpcoesMotorista:

O frmOpcoesMotorista tem como função mostrar as opções do programa relacionadas ao motorista (Incluir, relatórios etc).

2.4.3.10 Opções para Motorista (frmIncluirMotorista)

Incluir novo motorista

Insira os dados para incluir um novo motorista

Nome:

CPF:

Data de nascimento:

	Nome	CPF	Número CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111.111.111-11	06540403721	12	26/06/1998	Ativa
	Igor Gabriel	222.222.222-22	31506423546	3	26/06/1998	Ativa
	Igor Palhares	333.333.333-33	37044121240	7	26/06/1998	Ativa
	Guilherme	444.444.444-44	51360416108	5	26/06/1998	Ativa
	João Vitor	555.555.555-55	81720108153	7	26/06/1998	Ativa
	Gabriel Silva	666.666.666-66	58242038085	0	26/06/1998	Ativa
	Maria Silva	777.777.777-77	85083881815	0	26/06/1998	Ativa
	Pedro Henrique	888.888.888-88	23524024807	0	26/06/1998	Ativa
*						

2.4.3.10.1 Código do frmIncluirMotorista:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmIncluirMotorista : Form {
        public frmIncluirMotorista() {
            InitializeComponent();
            CarregarGridMotoristas();
        }

        private void LimparCampos() {
            txtNome.Clear();
        }
    }
}
```

```

        txtCpf.Clear();
    }

    private void btnIncluirMot_Click(object sender, EventArgs e) { //Botao para
incluir motoristas

        string nome, cpf, dataN, numCarteira;
        Motorista resultadoMotorista = null;

        if (txtNome.Text == "" || txtCpf.Text == "" ||
dateTimePicker.Value.ToString("dd/MM/yyyy") == "" || !txtCpf.MaskCompleted)
{ //Verifica se os campos foram inseridos corretamente
    MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
} else {
    try {
        nome = txtNome.Text;
        cpf = txtCpf.Text;
        dataN = dateTimePicker.Value.ToString("dd/MM/yyyy");

        resultadoMotorista = Controle.ListaMotoristas.Find(x => x.Cpf == cpf); //
Verifica se o CPF já existe

        if (resultadoMotorista == null) {
            numCarteira = GerarNumCNH(); //Gera um numero aleatório para a
CNH

            Motorista m = new Motorista(nome, numCarteira, 0, cpf, dataN, true);
// Cadastrar novo motorista começa com 0 pontos inicialmente
            Controle.ListaMotoristas.Add(m); //Adiciona o motorista na lista
            dtgMotoristas.Rows.Clear();
            CarregarGridMotoristas();
            LimparCampos();
            MessageBox.Show("Motorista criado com sucesso.\n\n|-----
Dados-----|\nNome: " + nome + "\nCPF: " + cpf + "\nData de nascimento: " +
dataN + "\nNúmero da CNH: " + numCarteira, "Sucesso", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        } else {
            MessageBox.Show("CPF já cadastrado no sistema.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }

    } catch (FormatException) { //Caso algum valor seja inserido
incorretamente
        MessageBox.Show("Insira apenas valores válidos.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
    } catch (Exception ex) { //Caso algum erro inesperado ocorra o programa
não irá interromper

```

```

        MessageBox.Show("Ocorreu um erro inesperado.\n" + ex, "ERROR",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private string GerarNumCNH() { //Gerar numero para a CNH do novo motorista
    string numCNH = "";
    bool existe = false;
    Random randNum = new Random();

    do {

        existe = false;
        for (int i = 0; i < 11; i++) {
            numCNH += randNum.Next(0, 9);
        }

        foreach (Motorista mot in Controle.ListaMotoristas) {
            if (mot.NumCarteira == numCNH) existe = true;
        }

    } while (existe == true);

    return numCNH;
}

private void CarregarGridMotoristas() {
    foreach (Motorista mot in Controle.ListaMotoristas) {
        if (mot.PodeDirigir == true) {
            dtgMotoristas.Rows.Add(mot.Nome,      mot.Cpf,      mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Ativa");
        } else {
            dtgMotoristas.Rows.Add(mot.Nome,      mot.Cpf,      mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Suspensa");
        }
    }
}

private void dtgMotoristas_CellContentClick(object sender,
DataGridViewCellEventArgs e) {

}

private void txtCpf_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {

}

```

```

private void frmIncluirMotorista_Load(object sender, EventArgs e)
{
}
}
}

```

2.4.3.10.2 Explicação do Código frmIncluirMotorista:

O form de incluir usuário tem a função de mostrar no DataGridView os motoristas já existentes e oferecer a opção de incluir novos motoristas, deixando para que seja preenchido as informações do motorista e gerando um numero de CNH random (de forma que nunca se repita os números de CHN já existentes no sistema).

2.4.3.11 Relatórios de Motoristas (frmRelatorioMotoristas):

Relatório de motoristas

Escolha o filtro para o relatório de motoristas

☒ Todos
 ☐ Possuem carteira ativa
 ☐ Não possuem multas

Pesquisar motorista por CPF:

	Nome	CPF	Número da CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111,111,111-11	06540403721	12	26/06/1998	Ativa
	Igor Gabriel	222,222,222-22	31506423546	3	26/06/1998	Ativa
	Igor Palhares	333,333,333-33	37044121240	7	26/06/1998	Ativa
	Guilherme	444,444,444-44	51360416108	5	26/06/1998	Ativa
	João Vitor	555,555,555-55	81720108153	7	26/06/1998	Ativa
	Gabriel Silva	666,666,666-66	58242038085	0	26/06/1998	Ativa
	Maria Silva	777,777,777-77	85083881815	0	26/06/1998	Ativa
	Pedro Henrique	888,888,888-88	23524024807	0	26/06/1998	Ativa
*						

2.4.3.11.1 Códigos do frmRelatorioMotoristas:

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;

```



```

using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmRelatorioMotoristas : Form {
        public frmRelatorioMotoristas() {
            InitializeComponent();
            rbtnTodos.Checked = true;
        }

        private void CarregarGridMotoristas() {
            foreach (Motorista mot in Controle.ListaMotoristas) {
                if (mot.PodeDirigir == true) {
                    dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Ativa");
                } else {
                    dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Suspensa");
                }
            }
        }

        private void CarregarGridPossuemCarteira() {
            foreach (Motorista mot in Controle.ListaMotoristas) {
                if (mot.PodeDirigir == true) {
                    dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Ativa");
                }
            }
        }

        private void CarregarGridNaoMultados() {
            foreach (Motorista mot in Controle.ListaMotoristas) {
                if (mot.PontuacaoCarteira == 0) {
                    dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Ativa");
                }
            }
        }

        private bool CarregarGridCPF(string cpf) {
            bool achou = false;
            dtgMotoristas.Rows.Clear();
            foreach (Motorista mot in Controle.ListaMotoristas) {
                if (mot.Cpf == cpf) {

```

```

        achou = true;
        dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Ativa");
    }
}
return achou;
}

private void rbtnTodos_CheckedChanged(object sender, EventArgs e) {
    dtgMotoristas.Rows.Clear();
    CarregarGridMotoristas();
}

private void rbtnPossuemCarteira_CheckedChanged(object sender, EventArgs
e) {
    dtgMotoristas.Rows.Clear();
    CarregarGridPossuemCarteira();
}

private void rbtnNaoPossuemMultas_CheckedChanged(object sender,
EventArgs e) {
    dtgMotoristas.Rows.Clear();
    CarregarGridNaoMultados();
}

private void btnPesquisarCPF_Click(object sender, EventArgs e) {
    string cpf;
    rbtnTodos.Checked = false;

    cpf = txtCPF.Text;

    if (cpf == "" || !txtCPF.MaskCompleted) {
        MessageBox.Show("Para filtrar por CPF, digite o CPF desejado
corretamente.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        rbtnTodos.Checked = true;
    } else {
        if (!CarregarGridCPF(cpf)) {
            MessageBox.Show("CPF incorreto ou não existente.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            rbtnTodos.Checked = true;
        }
    }
}

private void frmRelatorioMotoristas_Load(object sender, EventArgs e)
{
}

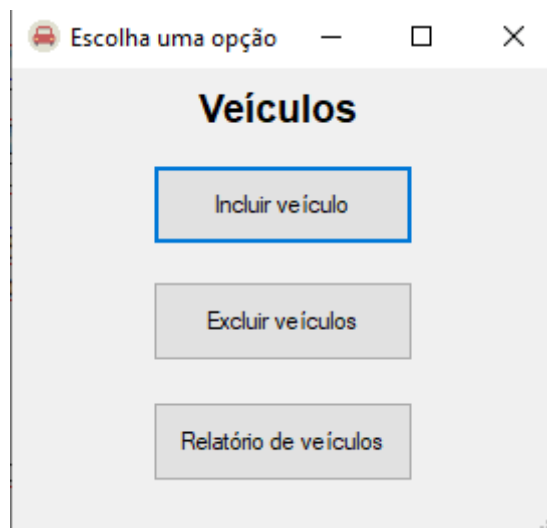
```

```
}  
}
```

2.4.3.11.2 Explicação códigos do frmRelatorioMotoristas:

O form de relatório de motoristas possui todos os motoristas cadastrados no sistema em um DataGrid, alguns filtros (procurar por CPF, apenas os que tem carteira ativa e etc), com a simples função de ser um relatório para conferência dos registros.

2.4.3.12 Opcoes Veículo (frmOpcoesVeiculo)



2.4.3.12.1 Código Opcoes Veículo (frmOpcoesVeiculo)

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;  
  
namespace TFI {  
    public partial class frmOpcoesVeiculo : Form {  
        public frmOpcoesVeiculo() {  
            InitializeComponent();  
        }  
    }  
}
```

```

    }

    private void btnIncluirVeiculo_Click(object sender, EventArgs e) {
        frmIncluirVeic incluirVeic = new frmIncluirVeic();
        incluirVeic.ShowDialog();
    }

    private void btnExcluirVeiculos_Click(object sender, EventArgs e) {
        frmExcluirVeic excluirVeic = new frmExcluirVeic();
        excluirVeic.ShowDialog();
    }

    private void btnRelatorioVeiculos_Click(object sender, EventArgs e) {
        frmRelatorioVeiculos relatVeic = new frmRelatorioVeiculos();
        relatVeic.ShowDialog();
    }


    private void frmOpcoesVeiculo_Load(object sender, EventArgs e)
    {
    }
}

```

2.4.3.12.2 Explicação Código Opções Veículo **(frmOpcoesVeiculo)**

O frmOpcoesVeiculo tem como função mostrar as opções do programa relacionadas a um veículo (Incluir, excluir, relatórios etc).

2.4.3.13 Incluir Veículo (frmIncluirVeiculo):

 Incluir novo veículo

Insira os dados para incluir um novo veículo

Placa: Gerar placa CPF do proprietário:

Chassi:

Modelo:

Ano:

Cadastrar veículo

Motoristas cadastrados

	Nome	CPF	Número da CNH	Pontuação	Data de nascimento	Carteira
▶	Paulo Rocha	111,111,111-11	06540403721	12	26/06/1998	Possui
	Igor Gabriel	222,222,222-22	31506423546	3	26/06/1998	Possui
	Igor Palhares	333,333,333-33	37044121240	7	26/06/1998	Possui
	Guilheme	444,444,444-44	51360416108	5	26/06/1998	Possui
	João Vitor	555,555,555-55	81720108153	7	26/06/1998	Possui
	Gabriel Silva	666,666,666-66	58242038085	0	26/06/1998	Possui
	Maria Silva	777,777,777-77	85083881815	0	26/06/1998	Possui
	Pedro Henrique	888,888,888-88	23524024807	0	26/06/1998	Possui
*						

2.4.3.13.1 Código do frmIncluirVeiculo:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmIncluirVeic : Form {
```

```

public frmIncluirVeic() {
    InitializeComponent();
    CarregarGridMotoristas();
}

private void LimparCampos() {
    txtPlaca.Clear();
    txtChassi.Clear();
    txtModelo.Clear();
    txtAno.Clear();
    txtCpfProprietario.Clear();
}

private void btnIncluirVeic_Click(object sender, EventArgs e) { //Botao para
incluir um veiculo
    string placa, modelo, cpfProprietario, chassi;
    int ano;

    Veiculo resultadoVeiculo = null, resultadoChassi = null;
    Motorista resultadoMotorista;

    try {
        if (txtPlaca.Text == "" || txtChassi.Text == "" || txtModelo.Text == "" ||
txtAno.Text == "" || txtCpfProprietario.Text == "" || !txtPlaca.MaskCompleted
|| !txtCpfProprietario.MaskCompleted) { //Verifica se os campos foram inseridos
corretamente
            MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        } else {
            placa = txtPlaca.Text;
            chassi = txtChassi.Text;
            modelo = txtModelo.Text;
            ano = int.Parse(txtAno.Text);
            cpfProprietario = txtCpfProprietario.Text;

            foreach (Motorista mot in Controle.ListaMotoristas) { // para cara
motorista o foreach verifica se já existe a placa
                resultadoVeiculo = mot.ListaVeiculos1.Find(x => x.Placa == placa);
                resultadoChassi = mot.ListaVeiculos1.Find(x => x.Chassi == chassi);
                if (resultadoVeiculo != null) {
                    MessageBox.Show("Veiculo já existente, insira outra placa.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return; // cancela a operação caso encontre um veiculo com a
mesma placa
                }

                if (resultadoChassi != null) {

```

```

        MessageBox.Show("Chassi já existente, insira outro.", "ERROR",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return; // cancela a operação caso encontre um veiculo com a
        mesma placa
    }
}

        resultadoMotorista = Controle.ListaMotoristas.Find(x => x.Cpf ==
        cpfProprietario); // verifica se o motorista que será proprietario existe

        if (resultadoMotorista != null) {

            Veiculo NovoVeic = new Veiculo(placa, chassi, modelo, ano,
            cpfProprietario, false);

            resultadoMotorista.ListaVeiculos1.Add(NovoVeic); //Adiciona o veiculo
            na lista

            LimparCampos();
            MessageBox.Show("Veiculo com a placa: " + placa + " cadastrado
            com sucesso.", "Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information);

        } else {
            MessageBox.Show("Motorista não existente ou CPF inválido, insira
            novamente.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    } catch (FormatException) { //Caso algum valor seja sido inserido com valores
    invalidos
        MessageBox.Show("Formato inválido, insira os caracteres corretamente.",
        "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
    } catch (Exception) { //Caso algum erro inesperado ocorra o programa não irá
    interromper
        MessageBox.Show("Ocorreu um erro inesperado, verifique com o
        administrador do sistema.", "ERROR", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    } finally {
        dtgMotoristas.Rows.Clear();
        CarregarGridMotoristas();
    }
}

private void CarregarGridMotoristas() {
    foreach (Motorista mot in Controle.ListaMotoristas) {
        if (mot.PodeDirigir == true) {
            dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
            mot.PontuacaoCarteira, mot.DataNasc, "Possui");
        } else {

```

```

        dtgMotoristas.Rows.Add(mot.Nome, mot.Cpf, mot.NumCarteira,
mot.PontuacaoCarteira, mot.DataNasc, "Suspensa");
    }
}

private void GerarPlacaRandom() { //Gerar uma placa random disponivel
    Random randNum = new Random();
    Veiculo veic;
    string placa = "";
    char letra;

    do {

        veic = null;
        placa = "";

        for (int i = 0; i < 3; i++) {
            letra = (char)randNum.Next(65, 90);
            placa += letra;
        }

        for (int i = 0; i < 4; i++) {
            placa += randNum.Next(0, 9);
        }

        foreach (Motorista mot in Controle.ListaMotoristas) {
            veic = mot.ListaVeiculos1.Find(x => x.Placa == placa);
        }

    } while (veic != null);

    txtPlaca.Text = placa;
}

private void txtPlaca_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {

}

private void btnGerarPlaca_Click(object sender, EventArgs e) { //Botao para
gerar uma placa random disponivel
    GerarPlacaRandom();
}

private void frmIncluirVeic_Load(object sender, EventArgs e)
{
}
}
}
}

```


2.4.3.13.2 Explicação código frmIncluirVeiculo:

Esse form tem como função a inclusão de um novo veículo a partir de dados inseridos, ele checa se há uma placa já existente para permitir ou não sua criação, também checa se o CPF do motorista que será o proprietário existe, possui também, um botão para gerar uma placa random que esteja disponível para uso, dessa forma o usuário pode escolher uma placa ou gerar de forma randômica. Possui também um DataGrid com os motoristas existentes para facilitar o cadastro do veículo na hora de inserir o CPF do proprietário.

2.4.3.14 Excluir Veículo (frmExcluirVeiculo):

Excluir um veículo

Digite a placa do veículo a ser excluído:

Excluir veículo

	Placa	Chassi	Modelo	Ano	Nome proprietário	CPF Proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha	111,111,111-11
	ESR-4221	34567	Gol	2014	Paulo Rocha	111,111,111-11
	GLB-6474	432566	Jeep	2016	Igor Gabriel	222,222,222-22
	WGM-5722	423432	Jeep	2019	Igor Palhares	333,333,333-33
	FLB-4260	3245263	Palio	2012	Guilherme	444,444,444-44
	CYQ-4504	3452687	Fox	2016	Gabriel Silva	666,666,666-66
	VRQ-5874	23563	Uno	2017	Maria Silva	777,777,777-77
*						

2.4.3.14.1 Códigos do frmExcluirVeiculo:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;
```

```

namespace TFI {
    public partial class frmExcluirVeic : Form {
        public frmExcluirVeic() {
            InitializeComponent();
            CarregarGridVeiculos(); //Carregar o grid de veiculos
        }

        private void LimparCampos() { //Limpar os campos
            txtPlacaEx.Clear();
        }

        private void btnExcluirVeic_Click(object sender, EventArgs e) { //Botao para
excluir um veiculo
            string placa;
            Veiculo resultVeiculo;

            bool achou = false;

            try {
                if (txtPlacaEx.Text == "" || !txtPlacaEx.MaskCompleted) { //Verifica se os
campos estão inseridos corretamente
                    MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
                } else {

                    placa = txtPlacaEx.Text;

                    foreach (Motorista mot in Controle.ListaMotoristas) { //Percorre a lista de
motoristas para percorrer os veiculos existentes
                        resultVeiculo = mot.ListaVeiculos1.Find(x => x.Placa == placa);
//Procura se o veiculo existe pela placa
                        if (resultVeiculo != null) { //Caso ache
                            achou = true;
                            if (resultVeiculo.Excluido == false) { //Se o veiculo já não tiver sido
excluido..
                                resultVeiculo.ExcluirVeic(); //Exclui o veiculo
                                dtgVeiculos.Rows.Clear(); //Limpa o grid
                                CarregarGridVeiculos(); //Carrega o grid de veiculos
                                LimparCampos(); //Limpa os campos
                                MessageBox.Show("Veiculo excluido com sucesso.", "Sucesso",
MessageButtons.OK, MessageBoxIcon.Information);
                            } else {
                                LimparCampos(); //Limpa os campos
                                MessageBox.Show("Esse veiculo já foi excluido.", "ERROR",
MessageButtons.OK, MessageBoxIcon.Error);
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

        if (achou == false) MessageBox.Show("Registro de veiculo não
encontrado, verifique a placa inserida.", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }

    } catch (NullReferenceException) { //Exceção para caso não ache o veiculo
        MessageBox.Show("Registro de veiculo não encontrado, verifique a placa
inserida.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
    } catch (Exception) { //Exceção geral para caso um erro inesperado ocorra
        MessageBox.Show("Ocorreu um erro inesperado, verifique com o
administrador do sistema.", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

private void CarregarGridVeiculos() {
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            if (v.Excluido == false) {
                dtgVeiculos.Rows.Add(v.Placa, v.Chassi, v.Modelo, v.Ano, mot.Nome,
v.CpfMotorista, "Não");
            }
        }
    }
}

private void dtgVeiculos_CellContentClick(object sender,
DataGridViewCellEventArgs e) {
    txtPlacaEx.Text = dtgVeiculos[0,
dtgVeiculos.CurrentCellAddress.Y].Value.ToString();
}

private void maskedTextBox1_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {
}

private void frmExcluirVeic_Load(object sender, EventArgs e)
{
}
}
}

```

2.4.3.14.2 Explicação do código do frmExcluirVeiculo:

O form para excluir um veículo possui como função apenas listar os veículos que não foram excluídos no DataGrid e a opção para excluir um veículo a partir da placa.

2.4.3.15 Relatórios Veículos (frmRelatorioVeiculos):

	Placa	Chassi	Modelo	Ano	Nome proprietário	CPF Proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha	111,111,111-11
	ESR-4221	34567	Gol	2014	Paulo Rocha	111,111,111-11
	GLB-6474	432566	Jeep	2016	Igor Gabriel	222,222,222-22
	WGM-5722	423432	Jeep	2019	Igor Palhares	333,333,333-33
	FLB-4260	3245263	Palio	2012	Guilherme	444,444,444-44
	CYQ-4504	3452687	Fox	2016	Gabriel Silva	666,666,666-66
	VRQ-5874	23563	Uno	2017	Maria Silva	777,777,777-77
*						

2.4.3.15.1 Códigos do frmRelatorioVeiculos:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Threading.Tasks;  
using System.Windows.Forms;
```

```
namespace TFI {
```

```

public partial class frmRelatorioVeiculos : Form {
    public frmRelatorioVeiculos() {
        InitializeComponent();
        rbtnTodos.Checked = true;
    }

    private void rbtnTodos_CheckedChanged(object sender, EventArgs e) {
        dtgVeiculos.Rows.Clear();
        CarregarGridTodosVeiculos();
    }

    private void CarregarGridTodosVeiculos() {
        foreach (Motorista mot in Controle.ListaMotoristas) {
            foreach (Veiculo v in mot.ListaVeiculos1) {
                if(v.Excluido == false) dtgVeiculos.Rows.Add(v.Placa, v.Chassi,
v.Modelo, v.Ano, mot.Nome, v.CpfMotorista, "Não");
            }
        }
    }

    private bool PesquisaPorCPF(string CPFProprietario) {
        bool achou = false;
        dtgVeiculos.Rows.Clear();
        foreach (Motorista mot in Controle.ListaMotoristas) {
            if (mot.Cpf == CPFProprietario) { //Caso ache o CPF inserido, irá imprimir
            todos os carros que o proprietario possui (caso possua)
                foreach (Veiculo v in mot.ListaVeiculos1) {
                    achou = true;
                    if (v.Excluido == false) dtgVeiculos.Rows.Add(v.Placa, v.Chassi,
v.Modelo, v.Ano, mot.Nome, v.CpfMotorista, "Não");
                }
            }
        }
        return achou;
    }

    private bool PesquisasPorPlaca(string placa) {
        bool achou = false;
        dtgVeiculos.Rows.Clear();
        foreach (Motorista mot in Controle.ListaMotoristas) {
            foreach (Veiculo v in mot.ListaVeiculos1) {
                if (v.Placa == placa) {
                    achou = true;
                    if (v.Excluido == false) dtgVeiculos.Rows.Add(v.Placa, v.Chassi,
v.Modelo, v.Ano, mot.Nome, v.CpfMotorista, "Não");
                    break;
                }
            }
        }
    }
}

```

```

        if (achou == true) break;
    }
    return achou;
}

private bool PesquisaPorPlacaCPF(string placa, string cpf) {
    bool achou = false;
    dtgVeiculos.Rows.Clear();
    foreach (Motorista mot in Controle.ListaMotoristas) {
        if (mot.Cpf == cpf) {
            foreach (Veiculo v in mot.ListaVeiculos1) {
                if (v.Placa == placa) {
                    achou = true;
                    if (v.Excluido == false) dtgVeiculos.Rows.Add(v.Placa, v.Chassi,
v.Modelo, v.Ano, mot.Nome, v.CpfMotorista, "Não");
                }
            }
        }
    }
    return achou;
}

private void btnFiltrarPesquisa_Click(object sender, EventArgs e) {
    string placa, cpf;

    rbtnTodos.Checked = false;

    if (!txtPlaca.MaskCompleted) placa = "";
    else placa = txtPlaca.Text;

    if (!txtCPFProprietario.MaskCompleted) cpf = "";
    else cpf = txtCPFProprietario.Text;

    if (placa == "" && cpf == "") { //Caso nenhum dos campos seja preenchido.
        MessageBox.Show("O(s) campo(s) precisa(m) ser preenchido(s)
corretamente para a filtragem.", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    } else if (placa == "") { //Filtrar apenas por CPF caso o campo de placa esteja
vazio.

        if (!PesquisaPorCPF(cpf)) {
            rbtnTodos.Checked = true;
            MessageBox.Show("Nenhum registro de veiculo foi encontrado para
essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

    } else if (cpf == "") { //Filtrar apenas por placa caso o campo de CPF esteja
vazio.

```

```

        if (!PesquisasPorPlaca(placa)) {
            rbtnTodos.Checked = true;
            MessageBox.Show("Nenhum registro de veiculo foi encontrado para
            essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

    } else { //Filtrar por CPF e por placa caso ambos estejam preenchidos.

        if (!PesquisaPorPlacaCPF(placa, cpf)) {
            rbtnTodos.Checked = true;
            MessageBox.Show("Nenhum registro de veiculo foi encontrado para
            essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
}

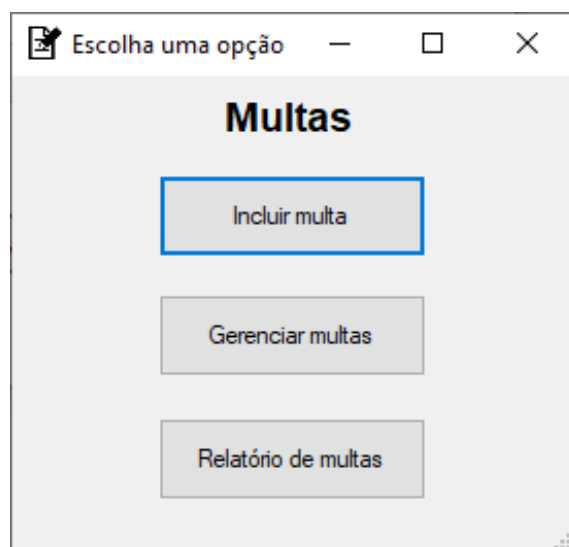
private void frmRelatorioVeiculos_Load(object sender, EventArgs e)
{
}
}
}

```

2.4.3.15.2 Explicação códigos do frmRelatorioVeiculos:

O form tem como função mostrar todos os relatórios de veículos cadastrados no sistema, possuindo alguns filtros (procurar por placa, mostrar todos os veículos de um motorista específico e etc) de forma que fique para conferência de registros.

2.4.3.16 Multas (frmOpcoesMulta)



2.4.3.16.1 Código Multas (frmOpcoesMulta):

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmOpcoesMulta : Form {
        public frmOpcoesMulta() {
            InitializeComponent();
        }

        private void btnIncluirMulta_Click(object sender, EventArgs e) {
            frmIncluirMulta incluirMulta = new frmIncluirMulta();
            incluirMulta.ShowDialog();
        }

        private void btnGerenciarMultas_Click(object sender, EventArgs e) {
            frmGerenciarMultas gerenciarMultas = new frmGerenciarMultas();
            gerenciarMultas.ShowDialog();
        }

        private void btnRelatorioMultas_Click(object sender, EventArgs e) {
            frmRelatorioMultas relatMultas = new frmRelatorioMultas();
            relatMultas.ShowDialog();
        }

        private void frmOpcoesMulta_Load(object sender, EventArgs e)
        {
        }
    }
}
```

2.4.3.16.2 Explicação Código Multas (frmOpcoesMulta):

O frmOpcoesMulta tem como função mostrar todas as opções e funcionalidades relacionadas a multas (Incluir, gerenciar, relatórios).

2.4.3.17 Incluir Multa (frmIncluirMulta):

Incluir uma nova multa

Insira os dados para incluir uma nova multa

Digite a placa: Motivo da multa:

Gravidade: CPF do motorista:

Data:

Valor da multa:

	Placa	Chassi	Modelo	Ano	Nome proprietário
▶	NGC-4271	532532	Uno	2012	Paulo Rocha
	ESR-4221	34567	Gol	2014	Paulo Rocha
	GLB-6474	432566	Jeep	2016	Igor Gabriel
	WGM-5722	423432	Jeep	2019	Igor Palhares
	FLB-4260	3245263	Palio	2012	Guilherme
	CYQ-4504	3452687	Fox	2016	Gabriel Silva
	VRQ-5874	23563	Uno	2017	Maria Silva
*					

2.4.3.17.1 Código do frmIncluirMulta:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmIncluirMulta : Form {
```

```

public frmIncluirMulta() {
    InitializeComponent();
    CarregarGridVeiculos();
    cbxGravidade.SelectedIndex = 0;
}

private void LimparCampos() {
    txtPlaca.Clear();
    txtCpfM.Clear();
    txtValorM.Clear();
    cbxGravidade.SelectedIndex = 0;
}

private void btnAplicarMulta_Click(object sender, EventArgs e) { //Botao para
aplicar multas
    int gravidade;
    string dataMulta, cpfMotorista, placa;
    double valor;

    Motorista resultadoMotorista;
    Veiculo resultadoVeiculo = null;

    try {
        if (txtPlaca.Text == "" || txtCpfM.Text == "" || txtValorM.Text == ""
        || !txtPlaca.MaskCompleted || !txtCpfM.MaskCompleted) { //Verifica se os campos
        foram inseridos corretamente
            MessageBox.Show("Insira todos os dados necessários corretamente.",
            "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        } else {
            placa = txtPlaca.Text;
            gravidade = cbxGravidade.SelectedIndex;
            dataMulta = dateTimePicker.Value.ToString("dd/MM/yyyy");
            cpfMotorista = txtCpfM.Text;
            valor = double.Parse(txtValorM.Text);

            resultadoMotorista = Controle.ListaMotoristas.Find(x => x.Cpf ==
            cpfMotorista); // verifica se o motorista a ser multado existe

            if (resultadoMotorista != null) {

                foreach (Motorista mot in Controle.ListaMotoristas) { // para cara
                motorista o foreach verifica se existe a placa inserida
                    resultadoVeiculo = mot.ListaVeiculos1.Find(x => x.Placa == placa);
                    if (resultadoVeiculo != null && resultadoVeiculo.Excluido == false)
                    { //Se achar o veiculo e ele nao estiver sido excluido, aplica a multa

                        Multa multa = new Multa(resultadoVeiculo.ListaMultas1.Count(),
                        gravidade, dataMulta, valor, cpfMotorista, false, false, false); //Criando a multa

```

```

        multa.CalcularPenalidade(gravidade); //Calcula o valor em
        pontos da penalidade dessa multa
        resultadoVeiculo.ListaMultas1.Add(multa); //Adiciona essa multa
        na lista de multas do veiculo

        resultadoMotorista.AplicarPenalidade(multa.Penalidade); //Aplica
        a penalidade dos pontos no motorista multado
        LimparCampos();
        MessageBox.Show("Multa cadastrada com sucesso.", "Sucesso",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
        return; // termina o loop pois não precisa mais continuar
    }
}
if (resultadoVeiculo == null || resultadoVeiculo.Excluido == true) {
    MessageBox.Show("Veiculo não encontrado ou já excluido,
    verifique se a placa foi inserida corretamente.", "ERROR", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
} else {
    MessageBox.Show("Motorista não existente ou CPF inválido, insira
    novamente.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

} catch (FormatException) { //Caso algum valor seja inserido incorretamente
    MessageBox.Show("Formato inválido, insira valores e caracteres válidos.",
    "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
} catch (Exception ex) { //Caso algum erro inesperado ocorra o programa não
    ira interromper
    MessageBox.Show("Ocorreu um erro inesperado.\n" + ex, "ERROR",
    MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}

private void CarregarGridVeiculos() {
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            if(v.Excluido == false) dtgVeiculos.Rows.Add(v.Placa, v.Chassi,
            v.Modelo, v.Ano, mot.Nome, v.CpfMotorista, "Não");
        }
    }
}

private void txtPlaca_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {
}

```

```

        private void txtCpfM_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {

    }

    private void frmIncluirMulta_Load(object sender, EventArgs e)
    {

    }

}
}
}

```

2.4.3.17.2 Explicação Código do frmIncluirMulta:

O código de incluir multas possui como função apenas incluir uma multa (informa-se a placa, gravidade, data da multa, valor e o CPF do multado, que pode ser o proprietário do carro ou não, e o motivo da multa), os métodos fazem a verificação se os dados estão inseridos corretamente e impossibilitam a inserção de valores inválidos, assim como todos os outros forms fazem.

2.4.3.18 Gerenciar Multas (frmGerenciarMultas):

Gerenciar multas

Veículos

Placa para verificar multas:

	Placa	Chassi	Modelo	Ano	Nome proprietário
	NGC-4271	532532	Uno	2012	Paulo Rocha
	ESR-4221	34567	Gol	2014	Paulo Rocha
▶	GLB-6474	432566	Jeep	2016	Igor Gabriel
	WGM-5722	423432	Jeep	2019	Igor Palhares
	FLB-4260	3245263	Palio	2012	Guilherme
	CYQ-4504	3452687	Fox	2016	Gabriel Silva
	VRQ-5874	23563	Uno	2017	Maria Silva
*					

Multas

☒ Quitar uma multa
 ☐ Transferir uma multa
 ☐ Justificar uma multa

ID da multa:

Placa:

CPF do Multado:

	ID da multa	Placa	Motivo	Paga	Gravidade
▶	1	GLB-6474	Estacionar em lo...	Não	Leve
	2	GLB-6474	Ultrapassar faixa ...	Não	Gravíssima
*					

2.4.3.18.1 Código do frmGerenciarMultas:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmGerenciarMultas : Form {
        public frmGerenciarMultas() {
            InitializeComponent();
            CarregarGridVeiculos();
            rbtnQuitar.Checked = false;
            rbtnTransferir.Checked = false;
            rbtnJustificar.Checked = false;
        }

        private void btnVerificar_Click(object sender, EventArgs e) { //Botao para
            verificar multas de um veiculo a partir da placa

            dtgMultas.Rows.Clear();

            string placa = txtPlaca.Text;

            if (!txtPlaca.MaskCompleted) {
                MessageBox.Show("Insira todos os dados necessários corretamente.",
                "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
            } else {
                Veiculo resultadoVeiculo = null;
            }
        }
    }
}
```

```

        foreach (Motorista mot in Controle.ListaMotoristas) { // para cara motorista
o foreach verifica se existe a placa inserida
            resultadoVeiculo = mot.ListaVeiculos1.Find(x => x.Placa == placa);
//Verifica se a placa existe

            if (resultadoVeiculo != null) { //Caso exista ele carrega as multas daquele
veiculo, mas se o veiculo ja tiver sido excluido, ele avisa que os registros continuam.
                if (resultadoVeiculo.Excluido == true) MessageBox.Show("Esse
veiculo já foi excluido do sistema, porem as multas existentes no veiculo continuam
normalmente.", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);

                CarregarMultasVeiculo(resultadoVeiculo); //Carrega as multas de um
veiculo especifico

                return;
            }
        }

        if (resultadoVeiculo == null) {
            MessageBox.Show("Registro de veiculo não encontrado, verifique a
placa inserida.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

}

private void CarregarGridVeiculos() {
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            if (v.Excluido == false) {
                dtgVeiculos.Rows.Add(v.Placa, v.Chassi, v.Modelo, v.Ano, mot.Nome,
v.CpfMotorista, "Não");
            } else {
                dtgVeiculos.Rows.Add(v.Placa, v.Chassi, v.Modelo, v.Ano, mot.Nome,
v.CpfMotorista, "Sim");
            }
        }
    }
}

```

```

    }
}

```

```

private void CarregarMultasVeiculo(Veiculo veic) {
    string paga = "", transferida = "", justificada = "", gravidade = "";
    foreach (Multa m in veic.ListaMultas1) {
        if (m.Quitada == true) paga = "Sim";
        else paga = "Não";

        if (m.Transferida == true) transferida = "Sim";
        else transferida = "Não";

        if (m.Justificada == true) justificada = "Sim";
        else justificada = "Não";

        if (m.Gravidade == 0) gravidade = "Leve"; // leve
        else if (m.Gravidade == 1) gravidade = "Média"; // media
        else if (m.Gravidade == 2) gravidade = "Grave"; // grave
        else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
        else gravidade = "";

        if (m.Justificada == false) {
            dtgMultas.Rows.Add(m.Id + 1, veic.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
        } else {
            dtgMultas.Rows.Add(m.Id + 1, veic.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
        }
    }
}
}

```

```

private void dataGridView2_CellContentClick(object sender,
DataGridViewCellEventArgs e) {

```

```

        /*this.txtPlacaQuitar.Text = dataGridView2[1,
dataGridView1.CurrentCellAddress.Y].Value.ToString();
        this.txtID.Text = dataGridView2[0,
dataGridView1.CurrentCellAddress.Y].Value.ToString();*/
    }

```

```

private void frmQuitarMulta_Load(object sender, EventArgs e) {

}

```

```

private void rbtnQuitar_CheckedChanged(object sender, EventArgs e) {
    if (rbtnQuitar.Checked == true) {
        lblID.Visible = true;
        lblPlaca.Visible = true;
        lblCPFMultado.Visible = true;
        txtID.Visible = true;
        txtPlacaQuitar.Visible = true;
        txtCPFMultado.Visible = true;
        btnQuitarMulta.Visible = true;
    } else {
        lblID.Visible = false;
        lblPlaca.Visible = false;
        lblCPFMultado.Visible = false;
        txtID.Visible = false;
        txtPlacaQuitar.Visible = false;
        txtCPFMultado.Visible = false;
        btnQuitarMulta.Visible = false;
    }
}

```

```

private void rbtnTransferir_CheckedChanged(object sender, EventArgs e) {
    if (rbtnTransferir.Checked == true) {
        lblIDT.Visible = true;
        lblPlacaT.Visible = true;
    }
}

```



```

        lblTransferirCPF.Visible = true;
        txtIDT.Visible = true;
        txtPlacaT.Visible = true;
        txtNovoCPF.Visible = true;
        btnTransferirMulta.Visible = true;
    } else {
        lblIDT.Visible = false;
        lblPlacaT.Visible = false;
        lblTransferirCPF.Visible = false;
        txtIDT.Visible = false;
        txtPlacaT.Visible = false;
        txtNovoCPF.Visible = false;
        btnTransferirMulta.Visible = false;
    }
}

```

```

private void rbtnJustificar_CheckedChanged(object sender, EventArgs e) {
    if (rbtnJustificar.Checked == true) {
        lblIDJ.Visible = true;
        lblPlacaJ.Visible = true;
        lblCPFMultadoJ.Visible = true;
        lblJustificacaoJ.Visible = true;
        txtIDJ.Visible = true;
        txtPlacaJ.Visible = true;
        txtCPFMultadoJ.Visible = true;
        txtJustificacao.Visible = true;
        btnJustificarMulta.Visible = true;
    } else {
        lblIDJ.Visible = false;
        lblPlacaJ.Visible = false;
        lblCPFMultadoJ.Visible = false;
        lblJustificacaoJ.Visible = false;
        txtIDJ.Visible = false;
    }
}

```

```

        txtPlacaJ.Visible = false;
        txtCPFMultadoJ.Visible = false;
        txtJustificacao.Visible = false;
        btnJustificarMulta.Visible = false;
    }
}

```

```

private void btnQuitarMulta_Click(object sender, EventArgs e) { //Botao para
quitar (pagar) multas

```

```

    int id;
    string placaQuitar, CPFMultado;

```

```

    bool achou = false;
    try {

```

```

        if (txtID.Text == "" || !txtPlacaQuitar.MaskCompleted
|| !txtCPFMultado.MaskCompleted) { //Verifica se todos os dados estão sendo
preenchidos corretamente

```

```

            MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

```

```

        } else {
            placaQuitar = txtPlacaQuitar.Text;
            id = int.Parse(txtID.Text) - 1;
            CPFMultado = txtCPFMultado.Text;

```

```

            foreach (Motorista mot in Controle.ListaMotoristas) { //Percorre as listas
para verificar se essa multa é valida

```

```

                foreach (Veiculo v in mot.ListaVeiculos1) {
                    if (placaQuitar == v.Placa) { //Verifica a placa do veiculo
                        foreach (Multa m in v.ListaMultas1) {

```

```

                            if (m.Id == id && m.CpfMultado == CPFMultado) { //Verifica o
CPF do motorista multado e o id da multa

```

```

                                achou = true;
                                if (m.Quitada == false && m.Justificada == false) {
                                    m.QuitarMulta(); //Quita a multa

```



```

Motorista resultadoMotorista;
Motorista ressarcido = null;

bool achou = false;

try {

    if (txtIDT.Text == "" || !txtPlacaT.MaskCompleted
|| !txtNovoCPF.MaskCompleted) { //Verifica se todos os campos foram inseridos
corretamente
        MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
    } else {
        placa = txtPlacaT.Text;
        id = int.Parse(txtIDT.Text) - 1;
        CPFTransferido = txtNovoCPF.Text;

        resultadoMotorista = Controle.ListaMotoristas.Find(x => x.Cpf ==
CPFTransferido); //Verificar se o CPF a ser transferido a multa existe
        if (resultadoMotorista == null) throw new NullReferenceException();

        foreach (Motorista mot in Controle.ListaMotoristas) { //Percorre as listas
para achar a multa
            foreach (Veiculo v in mot.ListaVeiculos1) {
                if (placa == v.Placa) {
                    foreach (Multa m in v.ListaMultas1) {
                        if (m.Id == id) {
                            achou = true;
                            if (m.Quitada == false && m.Transferida == false &&
m.Justificada == false) {
                                ressarcido = Controle.ListaMotoristas.Find(x => x.Cpf ==
m.CpfMultado); // Caso tudo dê certo, o motorista antigo terá os pontos ressarcidos
                                ressarcido.RessarcirPontos(m.Penalidade); //Motorista
com os pontos ressarcidos

```

como transferida

```
m.TransferirMulta(CPFTransferido); //Setando a multa
```

```
resultadoMotorista.AplicarPenalidade(m.Penalidade);  
//Penalizando o novo motorista responsavel pela multa
```

```
dtgMultas.Rows.Clear();  
CarregarMultasVeiculo(v);  
MessageBox.Show("Multa transferida com sucesso.",  
"Sucesso", MessageBoxButtons.OK, MessageBoxIcon.Information);  
} else {  
    MessageBox.Show("Essa multa já foi quitada e/ou  
transferida e/ou justificada, portanto não pode mais ser transferida.", "ERROR",  
MessageBoxButtons.OK, MessageBoxIcon.Error);  
}  
}  
}  
}  
}  
}
```

```
if (achou == false) MessageBox.Show("Multa não encontrada, verifique  
os dados novamente (ID, placa, CPF do multado).", "ERROR",  
MessageBoxButtons.OK, MessageBoxIcon.Error);  
}
```

```
} catch (FormatException) { //Caso algum valor seja inserido incorretamente  
    MessageBox.Show("Formato inválido, insira os caracteres corretamente.",  
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);  
}  
} catch (NullReferenceException) { //Caso a instancia do motorista não seja  
encontrado  
    MessageBox.Show("Motorista a ser transferido a multa não encontrado,  
verifique o CPF.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);  
}
```

```

        } catch (Exception) { //Caso ocorra um erro inesperado o programa não ira
interromper
        MessageBox.Show("Ocorreu um erro inesperado, verifique com o
administrador do sistema.", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
}

```

```

private void btnJustificarMulta_Click(object sender, EventArgs e) { //Botão de
justificar multa

```

```

    int id;
    string placa, CPFMultado, justificacao;
    Motorista motRessarcido;

```

```

    bool achou = false;

```

```

    try {

```

```

        if (txtIDJ.Text == "" || !txtPlacaJ.MaskCompleted
|| !txtCPFMultadoJ.MaskCompleted || txtJustificacao.Text == "") { //Verifica se todos
os campos foram inseridos corretamente

```

```

            MessageBox.Show("Insira todos os dados necessários corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);

```

```

        } else {

```

```

            placa = txtPlacaJ.Text;
            id = int.Parse(txtIDJ.Text) - 1;
            CPFMultado = txtCPFMultadoJ.Text;
            justificacao = txtJustificacao.Text;

```

```

            foreach (Motorista mot in Controle.ListaMotoristas) { //Percorre as listas
para achar a multa

```

```

                foreach (Veiculo v in mot.ListaVeiculos1) {

```

```

                    if (placa == v.Placa) {

```

```

                        foreach (Multa m in v.ListaMultas1) {

```

```

                            if (m.Id == id && m.CpfMultado == CPFMultado) {

```

```

        achou = true;
        if (m.Quitada == false && m.Transferida == false &&
m.Justificada == false) {

            motRessarcido = Controle.ListaMotoristas.Find(x =>
x.Cpf == m.CpfMultado); // Caso a multa seja justificada, o motorista terá os pontos
ressarcidos

            motRessarcido.RessarcirPontos(m.Penalidade);
//Ressarcindo os pontos

            m.JustificarMulta(justificacao); //Justifica a multa

            dtgMultas.Rows.Clear();
            CarregarMultasVeiculo(v);
            MessageBox.Show("Multa justificada com sucesso,
penalidade retirada.\nJustificação: " + justificacao, "Sucesso",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        } else {
            MessageBox.Show("Essa multa já foi quitada e/ou
transferida e/ou justificada, portanto não pode mais ser justificada.", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}

if (achou == false) MessageBox.Show("Multa não encontrada, verifique
os dados novamente (ID, placa, CPF do multado).", "ERROR",
MessageBoxButtons.OK, MessageBoxIcon.Error);
}

} catch (FormatException) { //Caso os valores sejam invalidos
    MessageBox.Show("Formato inválido, insira os caracteres corretamente.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

```

```

        } catch (NullReferenceException) { //Caso não encontre a referencia para
algum objeto
            MessageBox.Show("Um erro ocorreu, verifique os dados digitados.",
"ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        } catch (Exception) { //Caso algum erro inesperado ocorra o programa não
interrompe
            MessageBox.Show("Ocorreu um erro inesperado, verifique com o
administrador do sistema.", "ERROR", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
}

```

```

private void dtgVeiculos_CellContentClick(object sender,
DataGridViewCellEventArgs e) {
    dtgMultas.Rows.Clear();
    txtPlaca.Text = dtgVeiculos[0,
dtgVeiculos.CurrentCellAddress.Y].Value.ToString();
    Veiculo resultadoVeiculo = null;

    foreach (Motorista mot in Controle.ListaMotoristas) {
        resultadoVeiculo = mot.ListaVeiculos1.Find(x => x.Placa == txtPlaca.Text);

        if (resultadoVeiculo != null) {
            if (resultadoVeiculo.Excluido == true) MessageBox.Show("Esse veiculo
já foi excluido do sistema, porem as multas existentes no veiculo continuam
normalmente.", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            CarregarMultasVeiculo(resultadoVeiculo);
            return;
        }
    }
}

```

```

private void txtPlaca_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e) {

}

```



```

        private void dtgMultas_CellFormatting(object sender,
DataGridViewCellFormattingEventArgs e) {

    }

}
}
}

```

2.4.3.18.2 Explicação Código do frmGerenciarMultas:

O form de gerenciar multas tem como função mostrar os veículos em um DataGrid de forma que ele consiga verificar as multas que estão registradas naquele veículo, seja inserindo a placa e clicando no botão “Verificar” ou clicando na linha de registro do veículo no DataGrid, também tem como função quitar uma multa (pagar), transferir a multa ou justifica-lá, a partir das multas que são carregadas no DataGrid de multas, sempre conferindo dados inválidos e impossibilitando a ação caso os dados não sejam válidos. As opções (Quitar multa, transferir multa e justificar multa) são mostradas de acordo com o radio button selecionado.

2.4.3.19 Relatórios Multas (frmRelatorioMultas):

Relatório de multas

Escolha o filtro para o relatório de multas

☒ Todas
☐ Multas pagas
☐ Multas não pagas
☐ Multas transferidas
☐ Multas justificadas

Filtrar multas por placa:

Filtrar multas por CPF:

	Placa	Motivo	Paga	Gravidade	Data da multa	Valor	Penalidade	CPF do multado	Transferida	Justificada
▶	NGC-4271	Ultrapassar sinal ...	Não	Grave	06/06/2019	256	5	111,111,111-11	Não	Não
	GLB-6474	Estacionar em lo...	Não	Leve	06/06/2019	103	3	222,222,222-22	Não	Não
	GLB-6474	Ultrapassar faixa ...	Não	Gravíssima	07/06/2019	321	7	333,333,333-33	Não	Não
	WGM-5722	Ultrapassagem n...	Não	Gravíssima	06/06/2019	352	7	111,111,111-11	Não	Não
	CYQ-4504	Dirigir com farol d...	Não	Grave	06/06/2019	167	5	444,444,444-44	Não	Não
	VRQ-5874	Direção perigosa	Não	Gravíssima	06/06/2019	309	7	555,555,555-55	Não	Não

2.4.3.19.1 Códigos do frmRelatorioMultas:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace TFI {
    public partial class frmRelatorioMultas : Form {
        public frmRelatorioMultas() {
            InitializeComponent();
            rbtnTodas.Checked = true;
        }

        public void CarregarMultas() {
            string paga = "", transferida = "", justificada = "", gravidade = "";

            foreach (Motorista mot in Controle.ListaMotoristas) {
                foreach (Veiculo v in mot.ListaVeiculos1) {
                    foreach (Multa m in v.ListaMultas1) {
                        if (m.Quitada == true) paga = "Sim";
                        else paga = "Não";

                        if (m.Transferida == true) transferida = "Sim";
                        else transferida = "Não";

                        if (m.Justificada == true) justificada = "Sim";
                        else justificada = "Não";
                    }
                }
            }
        }
    }
}
```

```

        if (m.Gravidade == 0) gravidade = "Leve"; // leve
        else if (m.Gravidade == 1) gravidade = "Média"; // media
        else if (m.Gravidade == 2) gravidade = "Grave"; // grave
        else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
        else gravidade = "";

        if (m.Justificada == false) {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
        } else {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
        }
    }
}
}

public void CarregarMultasPagas() {
    string transferida = "", justificada = "", gravidade = "";

    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            foreach (Multa m in v.ListaMultas1) {
                if (m.Quitada == true) {

                    if (m.Transferida == true) transferida = "Sim";
                    else transferida = "Não";

                    if (m.Justificada == true) justificada = "Sim";
                    else justificada = "Não";

                    if (m.Gravidade == 0) gravidade = "Leve"; // leve

```

```

else if (m.Gravidade == 1) gravidade = "Média"; // media
else if (m.Gravidade == 2) gravidade = "Grave"; // grave
else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
else gravidade = "";

if (m.Justificada == false) {
    dtgMultas.Rows.Add(v.Placa, "Sim", gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
} else {
    dtgMultas.Rows.Add(v.Placa, "Sim", gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
}
}
}
}
}
}
}
}
}

```

```

public void CarregarMultasNaoPagas() {
    string transferida = "", justificada = "", gravidade = "";

    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            foreach (Multa m in v.ListaMultas1) {
                if (m.Quitada == false) {

                    if (m.Transferida == true) transferida = "Sim";
                    else transferida = "Não";

                    if (m.Justificada == true) justificada = "Sim";
                    else justificada = "Não";

                    if (m.Gravidade == 0) gravidade = "Leve"; // leve
                    else if (m.Gravidade == 1) gravidade = "Média"; // media

```

```

else if (m.Gravidade == 2) gravidade = "Grave"; // grave
else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
else gravidade = "";

if (m.Justificada == false) {
    dtgMultas.Rows.Add(v.Placa, "Não", gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
} else {
    dtgMultas.Rows.Add(v.Placa, "Não", gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
}
}
}
}
}
}
}
}

```

```

public void CarregarMultasTransferidas() {
    string paga = "", justificada = "", gravidade = "";

    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            foreach (Multa m in v.ListaMultas1) {
                if (m.Transferida == true) {

                    if (m.Quitada == true) paga = "Sim";
                    else paga = "Não";

                    if (m.Justificada == true) justificada = "Sim";
                    else justificada = "Não";

                    if (m.Gravidade == 0) gravidade = "Leve"; // leve
                    else if (m.Gravidade == 1) gravidade = "Média"; // media
                    else if (m.Gravidade == 2) gravidade = "Grave"; // grave

```



```

        else gravidade = "";

        dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, "Sim", m.Justificacao);
    }
}
}
}
}
}

```

```

public bool PesquisaPorCPF(string cpf) {
    string paga = "", transferida = "", justificada = "", gravidade = "";
    dtgMultas.Rows.Clear();
    bool achou = false;
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            foreach (Multa m in v.ListaMultas1) {
                if(m.CpfMultado == cpf) {
                    achou = true;
                    if (m.Quitada == true) paga = "Sim";
                    else paga = "Não";

                    if (m.Transferida == true) transferida = "Sim";
                    else transferida = "Não";

                    if (m.Justificada == true) justificada = "Sim";
                    else justificada = "Não";

                    if (m.Gravidade == 0) gravidade = "Leve"; // leve
                    else if (m.Gravidade == 1) gravidade = "Média"; // media
                    else if (m.Gravidade == 2) gravidade = "Grave"; // grave
                    else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
                    else gravidade = "";
                }
            }
        }
    }
    return achou;
}

```

```

        if (m.Justificada == false) {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
        } else {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
        }
    }
}
}
}
return achou;
}

```

```

public bool PesquisaPorPlaca(string placa) {
    string paga = "", transferida = "", justificada = "", gravidade = "";
    dtgMultas.Rows.Clear();
    bool achou = false;
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            if (v.Placa == placa) {
                achou = true;
                foreach (Multa m in v.ListaMultas1) {
                    if (m.Quitada == true) paga = "Sim";
                    else paga = "Não";

                    if (m.Transferida == true) transferida = "Sim";
                    else transferida = "Não";

                    if (m.Justificada == true) justificada = "Sim";
                    else justificada = "Não";

                    if (m.Gravidade == 0) gravidade = "Leve"; // leve

```



```

else if (m.Gravidade == 1) gravidade = "Média"; // media
else if (m.Gravidade == 2) gravidade = "Grave"; // grave
else if (m.Gravidade == 3) gravidade = "Gravíssima"; // gravissima
else gravidade = "";

if (m.Justificada == false) {
    dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
} else {
    dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
}
}
}
}
}
return achou;
}

```

```

public bool PesquisaPorPlacaCPF(string placa, string cpf) {
    string paga = "", transferida = "", justificada = "", gravidade = "";
    dtgMultas.Rows.Clear();
    bool achou = false;
    foreach (Motorista mot in Controle.ListaMotoristas) {
        foreach (Veiculo v in mot.ListaVeiculos1) {
            if (v.Placa == placa) {
                achou = true;
                foreach (Multa m in v.ListaMultas1) {
                    if (m.CpfMultado == cpf) {
                        if (m.Quitada == true) paga = "Sim";
                        else paga = "Não";

                        if (m.Transferida == true) transferida = "Sim";
                        else transferida = "Não";

```

```

        if (m.Justificada == true) justificada = "Sim";
        else justificada = "Não";

        if (m.Gravidade == 0) gravidade = "Leve"; // leve
        else if (m.Gravidade == 1) gravidade = "Média"; // media
        else if (m.Gravidade == 2) gravidade = "Grave"; // grave
        else if (m.Gravidade == 3) gravidade = "Gravíssima"; //
gravissima

        else gravidade = "";

        if (m.Justificada == false) {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, "Nenhuma");
        } else {
            dtgMultas.Rows.Add(v.Placa, paga, gravidade, m.DataMulta,
m.Valor, m.Penalidade, m.CpfMultado, transferida, justificada, m.Justificacao);
        }
    }
}
}
}
}
}
return achou;
}

private void rbtnTodas_CheckedChanged(object sender, EventArgs e) {
    dtgMultas.Rows.Clear();
    CarregarMultas();
}

private void rbtnPagas_CheckedChanged(object sender, EventArgs e) {
    dtgMultas.Rows.Clear();
    CarregarMultasPagas();
}

```

```

    }

    private void rbtnNaoPagas_CheckedChanged(object sender, EventArgs e) {
        dtgMultas.Rows.Clear();
        CarregarMultasNaoPagas();
    }

    private void rbtnTransferidas_CheckedChanged(object sender, EventArgs e) {
        dtgMultas.Rows.Clear();
        CarregarMultasTransferidas();
    }

    private void rbtnJustificadas_CheckedChanged(object sender, EventArgs e) {
        dtgMultas.Rows.Clear();
        CarregarMultasJustificadas();
    }

    private void btnFiltrar_Click(object sender, EventArgs e) {
        string placa, cpf;

        rbtnTodas.Checked = false;

        if (!txtPlacas.MaskCompleted) placa = "";
        else placa = txtPlacas.Text;

        if (!txtCPF.MaskCompleted) cpf = "";
        else cpf = txtCPF.Text;

        if (placa == "" && cpf == "") { //Caso nenhum dos campos seja preenchido.
            MessageBox.Show("O(s) campo(s) precisa(m) ser preenchido(s) corretamente para a filtragem.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        } else if (placa == "") { //Filtrar apenas por CPF caso o campo de placa esteja vazio.

```

```

        if (!PesquisaPorCPF(cpf)) {
            rbtnTodas.Checked = true;
            MessageBox.Show("Nenhum registro de multa foi encontrado para essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    } else if (cpf == "") { //Filtrar apenas por placa caso o campo de CPF esteja vazio.

        if (!PesquisaPorPlaca(placa)) {
            rbtnTodas.Checked = true;
            MessageBox.Show("Nenhum registro de multa foi encontrado para essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }

    } else { //Filtrar por CPF e por placa caso ambos estejam preenchidos.

        if (!PesquisaPorPlacaCPF(placa, cpf)) {
            rbtnTodas.Checked = true;
            MessageBox.Show("Nenhum registro de multa foi encontrado para essa pesquisa.", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        }
    }
}

private void frmRelatorioMultas_Load(object sender, EventArgs e)
{
}
}
}

```

2.4.3.19.2 Explicação códigos do frmRelatorioMultas:

O form de relatório de multas possui todas as multas cadastradas no sistema, para todos os veículos e motoristas e possui filtros para facilitar a busca (busca por CPF de um motorista, placa de veículo, multas justificadas, pagas e etc) o form tem como sua função principal apenas facilitar a conferência desses registros de multas.

3. CONCLUSÃO

Considerando as medidas e análises realizadas neste trabalho, percebemos que para a criação dos programas, foram utilizados recursos práticos e teóricos desenvolvidos em salas de aula e em laboratório de programação orientada a objeto como herança, polimorfismo, coesão, modularização, encapsulamento, UML, entre outros.

Neste trabalho, implementamos a UML (Unified Modeling Language) com o seu principal objetivo de especificar, construir, visualizar e documentar melhor os programas orientados a objetos.

Para a criação a UML, utilizamos o programa Astah UML que é um programa específico. E para a criação dos programas foram utilizados o Visual Studio e a linguagem de programação usada é o C#.

Conclui-se que nós aperfeiçoamos nossos conhecimentos para a criação de programas orientados a objeto e percebemos que este modelo de análise e programação é um método mais eficiente e mais empregado para o desenvolvimento de softwares.

REFERÊNCIAS

ARGOUML. In: **Wikipédia: a enciclopédia livre**. Disponível em: <<https://pt.wikipedia.org/wiki/ArgoUML>> Acesso em: 8 jun. 2019.

DETRAN. **Habilitação**. Disponível em: <<https://www.detran.mg.gov.br/>> Acesso em: 7 jun. 2019.

DEVMEDIA. **Programação Orientada a Objetos**. Disponível em: <<https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>> Acesso em: 4 jun. 2019.

INFOESCOLA. **Carteira Nacional de Habilitação (CNH)**. Disponível em: <<https://www.infoescola.com/transito/carteira-nacional-de-habilitacao-cnh/>> Acesso em: 8 jun. 2019.

SEABRA, Bruno. **O que é Astah?**. Disponível em: <<http://www.startupsstars.com/2015/10/o-que-e-o-astah-posttecnico-por-brunoseabra>> Acesso em: 8 jun. 2019.

Você sabe como funciona o sistema de pontos na CNH? Vamos explicar! Disponível em: <<https://doutormultas.jusbrasil.com.br/artigos/490127685/voce-sabe-como-funciona-o-sistema-de-pontos-na-cnh-vamos-explicar>>. Acesso em Maio 2019.

Carteira Nacional de Habilitação (CNH) possui cinco categorias. Disponível em: <<http://www.brasil.gov.br/cidadania-e-justica/2009/10/carteira-nacional-de-habilitacao-cnh-possui-cinco-categorias>> Acesso em Maio 2019.

Tirar sua 1ª habilitação, CNH, carteira de motorista? (Veja como). Disponível em: <<https://www.noticiasautomotivas.com.br/vai-tirar-sua-primeira-habilitacao-conheca-o-passo-a-passo/>> . Acesso em Maio 2019.

Como fazer o cadastro de veículo na BIN ? . Disponível em: <<https://www.consultaauto.com.br/blog/informativo/cadastro-de-veiculo-na-bin>>. Acesso em Maio 2019.