**TUTORIAL DE INSTALAÇÃO E PROGRAMAÇÃO EM PYHON**

1. **PYTHON**

A Linguagem Python foi concebida no fim dos anos 80 por Guido Van Rossum, enquanto trabalhava no [CWI](https://www.cwi.nl/) (Centrum Wiskunde & Informatica, Centro de Matemática e Ciência da Computação) no time de desenvolvimento da Linguagem ABC.

É uma linguagem expressiva, em que é fácil traduzir o raciocínio em um algoritmo. Em aplicações científicas, o raciocínio é essencialmente complicado. Um problema adicional para o cientista é ter que se preocupar com, além do assunto básico de sua pesquisa, a correção do programa em detalhes pouco relevantes. Python faz isso automaticamente de maneira muito eficiente, permitindo ao cientista se concentrar exclusivamente no problema sendo estudado.

1. **ESTUDO DE CASO**

O trabalho baseia-se em uma Locadora de Carros, em que o sistema é responsável pelo o cadastro de clientes, de veículos e controle de alugueis.

* Cadastro de Clientes e Veículos:
* Salva o cadastro;
* Listar Clientes, Veículos e Aluguéis:
  + Pesquisa e mostra informações;
  + Permite alterações;
  + Chama função para exclusão de cadastro.
* Alugar Veículo:
  + Vincula Cliente com veículo desejado.
  1. **DIAGRAMA DE CLASSE**

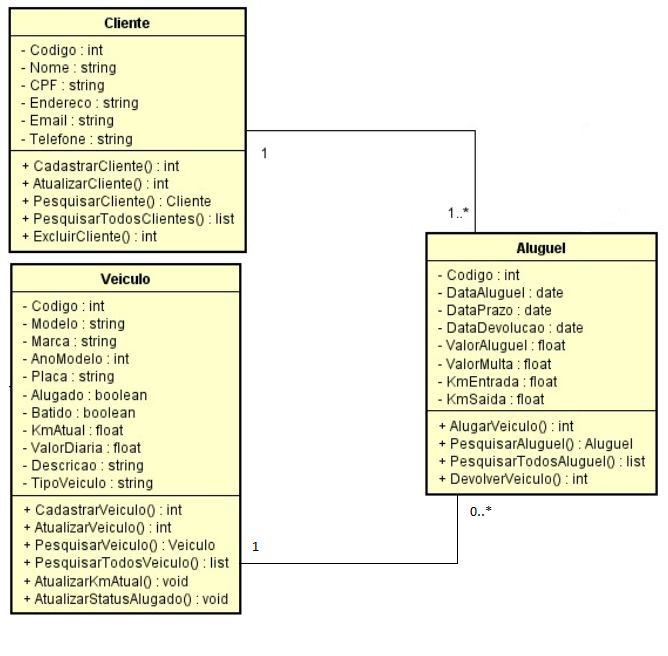


Figura 1. Diagrama de classe.

* 1. **ORGANIZAÇÃO DO PROGRAMA**

O programa será ordenado em camadas MVC (Model-View-Controller), composto por três camadas fundamentais, facilitando assim o desenvolvimento e permitindo manutenção da aplicação com agilidade.

MVC é uma forma de estruturar seu projeto/aplicação de forma que a interface de interação (**view**) esteja separada do controle da informação em si (**models**), separação essa que é intermediada por uma outra camada controladora (**controllers**).

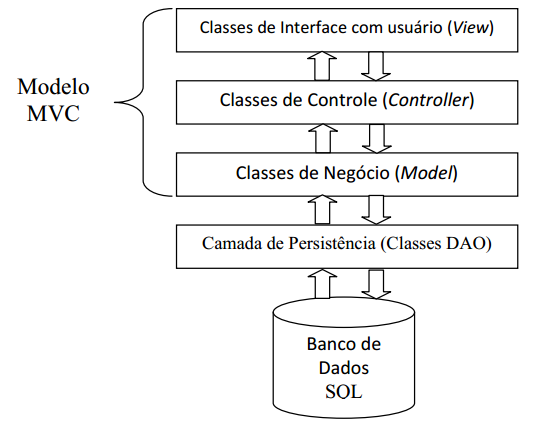


Figura 2. Modelo MVC.

1. **INSTALAÇÃO DO PYTHON E APLICAÇÕES AUXILIARES**
   1. **INSTALAÇÃO DO PYTHON**

Como a linguagem proposta para o desenvolvimento do trabalho é Python, será utilizada a versão para 32 bits do Python 3.4 como interpretador da linguagem. Pode ser baixado em: <https://www.python.org/downloads/release/python-340/>.

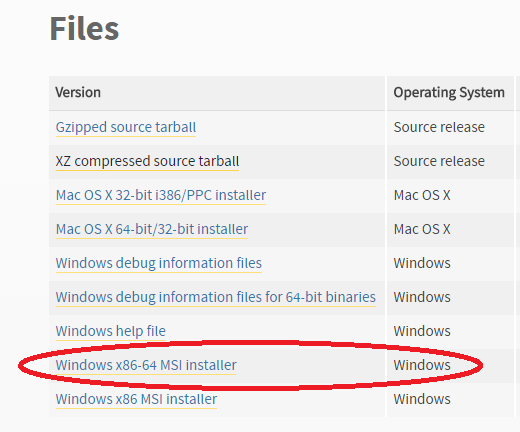


Figura 3. Download do Python para Windows.

* 1. **INSTALAÇÃO DO PYQT (BIBLIOTECA PARA INTERFACE GRÁFICA)**

Para a interface gráfica será utilizada a biblioteca PyQt4 que fornece com a instalação o Qt Designer, uma IDE para a customização das janelas, facilitando a implementação e economizando tempo. Pode ser baixado em:

<https://sourceforge.net/projects/pyqt/files/PyQt4/PyQt-4.11.4/PyQt4-4.11.4-gpl-Py3.4-Qt4.8.7-x32.exe/download>.

* 1. **INSTALAÇÃO DO CONTROLADOR DE BANCO DE DADOS (SQLite)**

Como controlador de banco de dados será utilizado o SQLite 3 por ser leve e simples de se usar. Pode ser baixado em:

<https://www.sqlite.org/download.html>

* 1. **INSTALAÇÃO DO DB BROWSER FOR SQLITE**

Para facilitar a visualização dos dados e a manipulação da estrutura do banco de dados, será utilizado o DB Browser for SQLite que pode ser baixado em:

[http://sqlitebrowser.org](http://sqlitebrowser.org/)

* 1. **INSTALAÇÃO DA IDE PYCHAM (AMBIENTE DE DESENVOLVIMENTO)**

Para facilitar a edição e interpretação dos códigos será utilizado o ambiente de desenvolvimento Pycham que pode ser baixado em:

<https://www.jetbrains.com/pycharm/>

1. **IMPLEMENTAÇÃO DO BANCO DE DADOS**

Foram criados três tabelas no banco de dados, seguindo o estudo de caso, conforme a Figura 4, 5 e 6.



Figura 4. Tabela Aluguel



Figura 5. Tabela Cliente



Figura 6. Tabela Veículo

1. **CLASSES DO MODELO, PROPRIEDADES E MÉTODOS**

Para iniciar o desenvolvimento do estudo de caso, foi criado um projeto chamado Projeto Pythom. Este projeto será divido em cinco diretórios como pode ser visualizado na Figura 7.

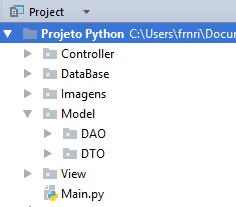


Figura 7. Diretórios do projeto do estudo de caso.

O diretório View, para armazenar as classes de interface, o Controller, para as classes controladoras, o Model, para as classes que contém as funções de banco de dados (DAO – Métodos; DTO - Atributos), a Imagens para todas as imagens utilizadas na interface, e, por fim, a DataBase para a conexão com o banco de dados.

* 1. **CONEXÃO COM O BANCO DE DADOS**

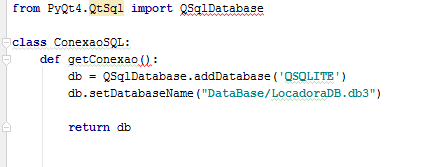


Figura 8. Implementação da conexão com o banco de dados.

É criada uma classe ConexaoSql que contém o método getConexao() que se conecta com o SQLITE e indica o caminho que está salvo o banco de dados.

Finalmente, retorna o objeto db (banco de dados).

* 1. **CADASTRO**

À partir da tela criada através da aplicação Qt Designer (Instalada juntamente com o PYQT), iniciamos a implementação.

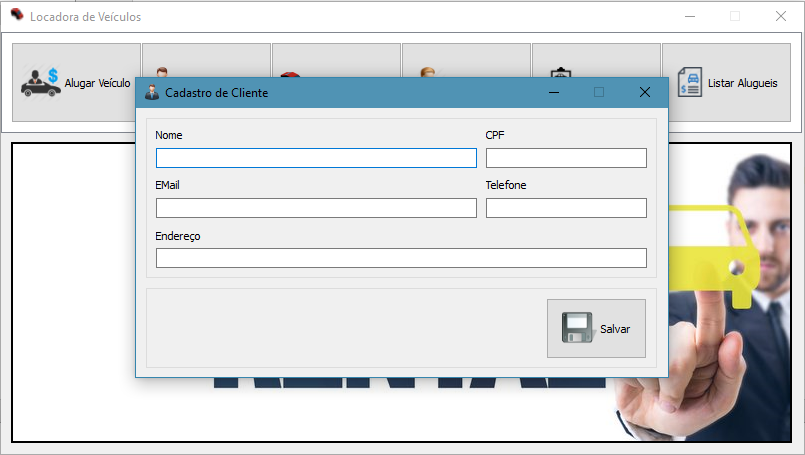


Figura 9. Tela Cadastro de Cliente.

Botão Salvar:

***#CLICK BTN\_SALVAR*def btnSalvar\_Click(self, estado, codigoCli):  
 nome = self.edtNome.text()  
 cpf = self.edtCPF.text()  
 endereco = self.edtEndereco.text()  
 email = self.edtEmail.text()  
 telefone = self.edtTelefone.text()**

***#VERIFICA O ESTADO INSERIR/ALTERAR PARA CHAMAR A FUNÇAO APROPRIADA* if estado=='inserir':  
 cliente = ClienteCTR  
 cliente.CadastrarCliente(nome, cpf, endereco, email, telefone)  
  
 if estado=='alterar':  
 cliente = ClienteCTR  
 cliente.AtualizarCliente(codigoCli, nome, cpf, endereco, email, telefone)**

* No evento salvar, são atribuídas para as variáveis as informações contidas nas respectivas caixas de texto;
* Faz uma verificação se está efetuando um cadastro ou alterando um cadastro já existente;
* Cria um objeto Cliente e salva as informações.

*\*Cadastro de Veículos e Aluguel funcionam da mesma forma.*

*\*Aluguel possui apenas a função de cadastro (sem alteração).*

* 1. **LISTAR**

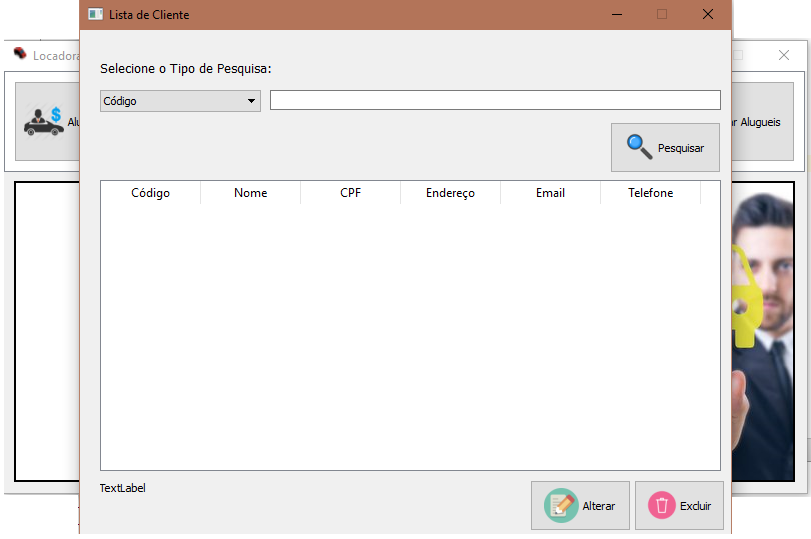


Figura 10. Tela Lista de Cliente.

Na tela de pesquisa, possibilita-se buscar um cliente desejado para alterar informações dele ou excluir o mesmo.

Botão Pesquisar:

**def PesquisarCliente(self, valor, tipo):  
 if valor == '':  
 self.PesquisarTodosClientes()  
 else:  
 cliente = ClienteCTR  
 query = cliente.PesquisarCliente(valor, tipo)  
  
 while (self.gridCliente.rowCount() > 0):  
 self.gridCliente.removeRow(0)  
 row = 0  
 while query.next():  
 self.gridCliente.insertRow(row)  
 codCli = QTableWidgetItem(str(query.value(0)))  
 nome = QTableWidgetItem(str(query.value(1)))  
 cpf = QTableWidgetItem(str(query.value(2)))  
 endereco = QTableWidgetItem(str(query.value(3)))  
 email = QTableWidgetItem(str(query.value(4)))  
 telefone = QTableWidgetItem(str(query.value(5)))  
  
 self.gridCliente.setItem(row, 0, codCli)  
 self.gridCliente.setItem(row, 1, nome)  
 self.gridCliente.setItem(row, 2, cpf)  
 self.gridCliente.setItem(row, 3, endereco)  
 self.gridCliente.setItem(row, 4, email)  
 self.gridCliente.setItem(row, 5, telefone)  
  
 row = row + 1  
  
def PesquisarTodosClientes(self):  
 cliente = ClienteCTR  
 query = cliente.PesquisarTodosClientes()  
  
 while (self.gridCliente.rowCount() > 0):  
 self.gridCliente.removeRow(0)  
  
 row = 0  
 while query.next():  
 self.gridCliente.insertRow(row)  
 codCli = QTableWidgetItem(str(query.value(0)))  
 nome = QTableWidgetItem(str(query.value(1)))  
 cpf = QTableWidgetItem(str(query.value(2)))  
 endereco = QTableWidgetItem(str(query.value(3)))  
 email = QTableWidgetItem(str(query.value(4)))  
 telefone = QTableWidgetItem(str(query.value(5)))  
  
 self.gridCliente.setItem(row, 0, codCli)  
 self.gridCliente.setItem(row, 1, nome)  
 self.gridCliente.setItem(row, 2, cpf)  
 self.gridCliente.setItem(row, 3, endereco)  
 self.gridCliente.setItem(row, 4, email)  
 self.gridCliente.setItem(row, 5, telefone)  
  
 row = row + 1**

* Quando o botão Pesquisa é acionado, primeiramente, verifica se há informações para filtrar a pesquisa;
* Se houver informação, é criado um objeto ClienteCTR;
* Chama a pesquisa no Banco de Dados (com as informações informadas);
* Limpa o Grid (Campo que aparece os clientes pesquisados);
* Insere as informações:
* Um cliente por linha;
* Uma informação por coluna, conforme especificado;
* Se não houver informação para filtrar o cliente, pesquisa todos eles (repetindo os passos acima).

Botão Excluir:

**def ExcluirCliente\_Click(self):  
  
 linha = self.gridCliente.currentItem().row()  
 codigoCli = self.gridCliente.item(linha, 0).text()  
  
 self.gridCliente.removeRow(linha)  
 cliente = ClienteCTR  
 cliente.ExcluirCliente(codigoCli)**

* Retira do Grid a linha do cliente selecionado para ser excluído;
* Cria um objeto ClienteCTR;
* Exclui cliente.

Botão Alterar:

**def AlterarCliente\_Click(self):  
 linha = self.gridCliente.currentItem().row()  
 codigoCli = self.gridCliente.item(linha, 0).text()  
 nome = self.gridCliente.item(linha, 1).text()  
 cpf = self.gridCliente.item(linha, 2).text()  
 endereco = self.gridCliente.item(linha, 3).text()  
 email = self.gridCliente.item(linha, 4).text()  
 telefone = self.gridCliente.item(linha, 5).text()  
  
 self.frmCliente = QtGui.QMainWindow()  
 self.ui = Ui\_frmCliente()  
 self.ui.setupUi(self.frmCliente, 'alterar', codigoCli)  
 self.ui.PreencherAlterar(nome, cpf, endereco, email, telefone)  
 self.frmCliente.show()**

* Salva informações geradas no Grid referente ao cliente nas respectivas variáveis;
* Chama tela de Cadastro de Cliente;
* Preenche os campos com as informações retiradas do Grid;
* Na tela Cadastro de Cliente (vista anteriormente), pode-se alterar as informações e salvá-las.

*\*Listar de Veículos e Aluguéis funcionam da mesma forma.*

* 1. **CLASSE CONTROLLER**

**class ClienteCTR:  
 def CadastrarCliente(nome, CPF, endereco, email, telefone):  
 clienteDTO = ClienteDTO  
 clienteDTO.Nome = nome  
 clienteDTO.CPF = CPF  
 clienteDTO.Endereco = endereco  
 clienteDTO.Email = email  
 clienteDTO.Telefone = telefone  
 clienteDAO = ClienteDAO  
 clienteDAO.CadastrarCliente(clienteDTO)  
  
 def AtualizarCliente(codigoCli, nome, CPF, endereco, email, telefone):  
  
 def PesquisarTodosClientes():  
  
 def PesquisarCliente(valor, tipo):  
  
 def ExcluirCliente(codigoCli):**

Após a chamada do método CadastrarCliente vindo do evento Salvar (da tela de Cadastro) é criado um objeto do tipo DTO (Classe que contém os atributos da Tabela de banco de dados):

* Atribui as informações contidas nas variáveis para o objeto ClienteDTO;
* É criado o objeto do tipo clienteDAO que acessa o método CadastrarCliente.

\**AtualizarCliente, PesquisarTodosClientes e ExcluirCliente funcionam da mesma maneira.*

* 1. **CLASSE MODEL**
     1. **DAO**

**class ClienteDAO:  
 def CadastrarCliente(cliente):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare("INSERT INTO Cliente(Nome, CPF, Endereco, Email, Telefone) "  
 "VALUES (?, ?, ?, ?, ?)")  
 query.addBindValue(cliente.Nome)  
 query.addBindValue(cliente.CPF)  
 query.addBindValue(cliente.Endereco)  
 query.addBindValue(cliente.Email)  
 query.addBindValue(cliente.Telefone)  
 query.exec\_()  
 db.commit()  
  
 def AtualizarCliente(codigoCli, cliente):  
   
 def ExcluirCliente(codigoCli):  
   
 def PesquisarTodosClientes():  
   
 def PesquisarCliente(valor, tipo):**

Na classe ClienteDao utiliza-se a conexão com o banco de dados para inserção das informações na tabela:

* Define o método CadastrarCliente para acessar a tabela Cliente;
* Query recebe a string SLQ para a inserção das informações nos atributos da tabela.

*\*Mesmo funcionamento das classes AluguelDAO e VeiculoDAO.*

* + 1. **DTO**

class ClienteDTO:  
 def \_\_init\_\_(self):  
 self.Nome = None  
 self.CPF = None  
 self.Endereco = None  
 self.Email = None  
 self.Telefone = None

* Cria o ClienteDTO com os atributos da classe.

*\*Mesmo funcionamento das classes AluguelDTO e VeiculoDTO.*

1. **CRIAÇÃO DE TELAS ATRAVÉS DO QT DESIGNER**

A ferramenta QT Designer foi utilizada para efetuar a parte visual deste projeto, a interação com o usuário.

Para utilizá-lo, basta selecionar o que deseja (Edits, botões, labels, etc) e arrastar para tela, montando-a como quiser. Segue abaixo visual do programa:

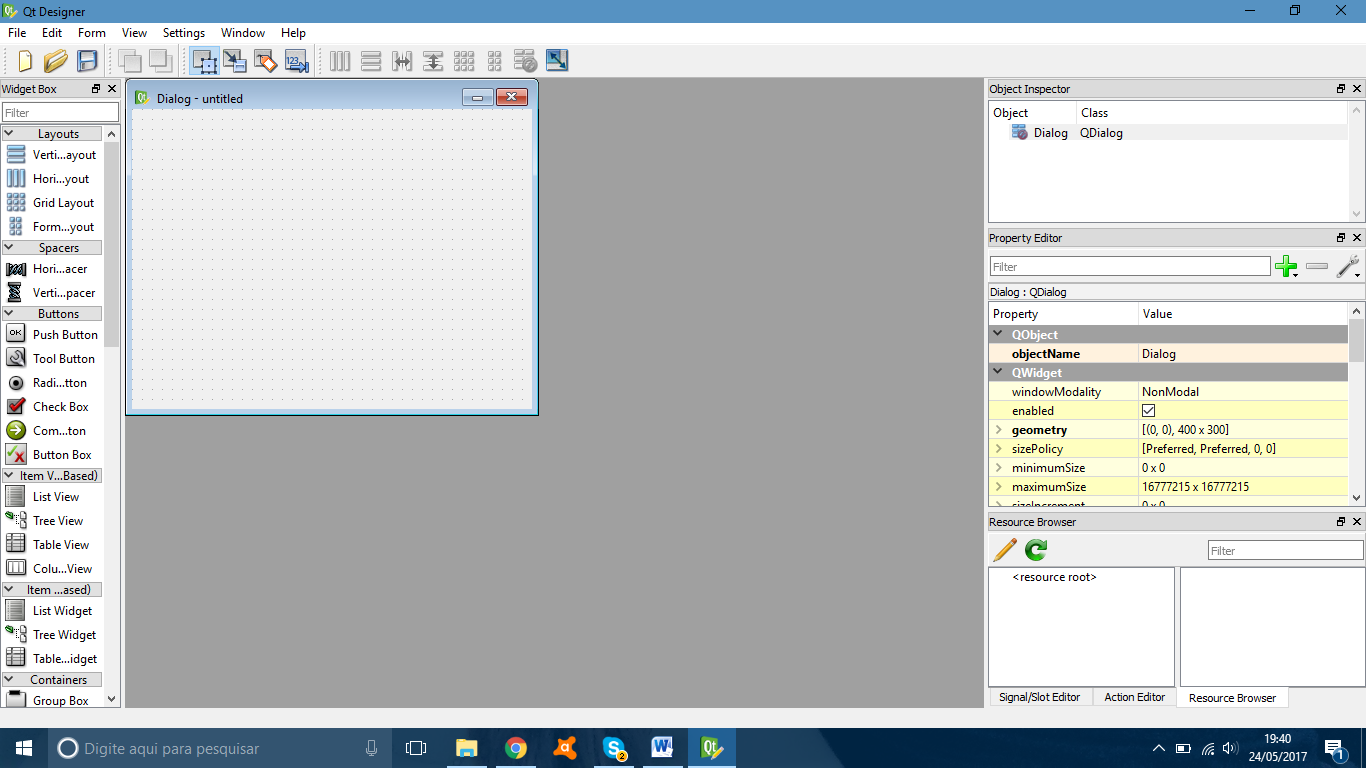


Figura 11. Criação de telas.

Porém, o QT Designer, permite apenas salvar a tela com uma extensão do tipo .UI e não em Python (.py). Por isso, faz-se necessário fazer a conversão das telas para podermos utilizá-las para programação em Python.

Para a conversão das telas, basta entrar na pasta em que foram salvas através do Prompt de Comando e digitar a seguinte instrução:

**pyuic4 -x FrmCliente.ui -o FrmCliente.py**

Nesse caso, estamos alterando a extensão do formulário de cadastro de clientes, mas, o mesmo se faz para todas as outras telas.

**APÊNDICE A – IMPLEMENTAÇÃO DAS CLASSES (DTO, DAO, CTR)**

**Cliente DTO**

**class** ClienteDTO:  
 **def** \_\_init\_\_(self):  
 self.Nome = **None** self.CPF = **None** self.Endereco = **None** self.Email = **None** self.Telefone = **None**

**Cliente DAO**

**class** ClienteDAO:  
 **def** CadastrarCliente(cliente):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"INSERT INTO Cliente(Nome, CPF, Endereco, Email, Telefone) VALUES (?, ?, ?, ?, ?)"**)  
 query.addBindValue(cliente.Nome)  
 query.addBindValue(cliente.CPF)  
 query.addBindValue(cliente.Endereco)  
 query.addBindValue(cliente.Email)  
 query.addBindValue(cliente.Telefone)  
 query.exec\_()  
 db.commit()  
  
 **def** AtualizarCliente(codigoCli, cliente):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"UPDATE Cliente SET Nome = '"**+cliente.Nome+**"', CPF = '"**+cliente.CPF+**"', Endereco = '"**+cliente.Endereco+**"', Email = "** +cliente.Email+**"', Telefone = '"**+cliente.Telefone+**"' WHERE CodigoCli ="** +codigoCli)  
 query.exec\_()  
 db.commit()  
  
 **def** ExcluirCliente(codigoCli):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"DELETE FROM Cliente WHERE CodigoCli=:codigoCli"**)  
 query.bindValue(**":codigoCli"**, codigoCli)  
 query.exec\_()  
 db.commit()

**def** PesquisarTodosClientes():  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 sql = **"SELECT \* FROM Cliente"** query = QSqlQuery(sql)  
  
 **return** query  
  
 **def** PesquisarCliente(valor, tipo):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 **if** tipo==**'Código'**:  
 sql = **"SELECT \* FROM Cliente where CodigoCli = "** + valor  
 query = QSqlQuery(sql)  
 **elif** tipo==**'Nome'**:  
 sql = **"SELECT \* FROM Cliente where Nome = '"**+valor+**"'"** query = QSqlQuery(sql)  
 **elif** tipo==**'CPF'**:  
 sql = **"SELECT \* FROM Cliente where CPF = '"** + valor+**"'"** query = QSqlQuery(sql)  
  
 **return** query  
  
  
**Cliente CTR**

**class** ClienteCTR:  
 **def** CadastrarCliente(nome, CPF, endereco, email, telefone):  
 clienteDTO = ClienteDTO  
 clienteDTO.Nome = nome  
 clienteDTO.CPF = CPF  
 clienteDTO.Endereco = endereco  
 clienteDTO.Email = email  
 clienteDTO.Telefone = telefone  
  
 clienteDAO = ClienteDAO  
 clienteDAO.CadastrarCliente(clienteDTO)  
  
 **def** AtualizarCliente(codigoCli, nome, CPF, endereco, email, telefone):  
 clienteDTO = ClienteDTO  
 clienteDTO.Nome = nome  
 clienteDTO.CPF = CPF  
 clienteDTO.Endereco = endereco  
 clienteDTO.Email = email  
 clienteDTO.Telefone = telefone  
  
 clienteDAO = ClienteDAO  
 clienteDAO.AtualizarCliente(codigoCli, clienteDTO)  
  
 **def** PesquisarTodosClientes():  
 clienteDAO = ClienteDAO  
 query = clienteDAO.PesquisarTodosClientes()  
  
 **return** query

**def** PesquisarCliente(valor, tipo):  
 clienteDAO = ClienteDAO  
 query = clienteDAO.PesquisarCliente(valor, tipo)  
  
 **return** query  
  
 **def** ExcluirCliente(codigoCli):  
 clienteDAO = ClienteDAO  
 clienteDAO.ExcluirCliente(codigoCli)

**Veiculo DTO**

**class** VeiculoDTO:  
 **def** \_\_init\_\_(self):  
 self.Modelo = **None** self.Marca = **None** self.AnoModelo = **None** self.Placa = **None** self.Alugado = **None** self.Batido = **None** self.KmAtual = **None** self.ValorDiaria = **None** self.Descricao = **None** self.TipoVeiculo = **None**

**Veiculo DAO**

**class** VeiculoDAO:  
 **def** CadastrarVeiculo(veiculo):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"INSERT INTO Veiculo(Modelo, Marca, AnoModelo, Placa, Alugado, Batido, KmAtual, ValorDiaria, Descricao, TipoVeiculo)VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"**)  
 query.addBindValue(veiculo.Modelo)  
 query.addBindValue(veiculo.Marca)  
 query.addBindValue(veiculo.AnoModelo)  
 query.addBindValue(veiculo.Placa)  
 query.addBindValue(veiculo.Alugado)  
 query.addBindValue(veiculo.Batido)  
 query.addBindValue(veiculo.KmAtual)  
 query.addBindValue(veiculo.ValorDiaria)  
 query.addBindValue(veiculo.Descricao)  
 query.addBindValue(veiculo.TipoVeiculo)  
  
 query.exec\_()  
 db.commit()

**def** AtualizarVeiculo(codigoVeic, veiculo):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"UPDATE Veiculo SET Modelo = '"**+veiculo.Modelo+**"', Marca = '"**+veiculo.Marca+**"', AnoModelo = "**+veiculo.AnoModelo+**", Placa = '"**+veiculo.Placa+**"', Alugado = '"**+veiculo.Alugado+**"', Batido = '"**+veiculo.Batido+**"', KmAtual = '"**+veiculo.KmAtual+**"', ValorDiaria = '"**+veiculo.ValorDiaria +**"', Descricao = '"**+veiculo.Descricao+**"', TipoVeiculo = '"**+veiculo.TipoVeiculo+**"' WHERE CodigoVeic = "**+codigoVeic)  
 query.exec\_()  
 db.commit()  
  
 **def** ExcluirVeiculo(codigoVeic):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
 query.prepare(**"DELETE FROM Veiculo WHERE CodigoVeic=:codigoVeic"**)  
 query.bindValue(**":codigoVeic"**, codigoVeic)  
 query.exec\_()  
 db.commit()  
  
 **def** PesquisarVeiculosDisponiveis():  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 sql = **"SELECT \* FROM Veiculo WHERE Alugado = 'Não'"** query = QSqlQuery(sql)  
  
 **return** query  
  
 **def** PesquisarTodosVeiculos():  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 sql = **"SELECT \* FROM Veiculo"** query = QSqlQuery(sql)  
  
 **return** query  
  
 **def** PesquisarVeiculo(valor, tipo):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 **if** tipo==**'Código'**:  
 sql = **"SELECT \* FROM Veiculo where CodigoVeic = "** + valor  
 query = QSqlQuery(sql)  
 **elif** tipo==**'Marca'**:  
 sql = **"SELECT \* FROM Veiculo where Marca = '"**+valor+**"'"** query = QSqlQuery(sql)  
 **elif** tipo==**'Modelo'**:  
 sql = **"SELECT \* FROM Veiculo where Modelo = '"** + valor+**"'"** query = QSqlQuery(sql)

**elif** tipo==**'Disponível'**:  
 sql = **"SELECT \* FROM Veiculo where Alugado = 'Não'"** query = QSqlQuery(sql)  
 **elif** tipo==**'Alugado'**:  
 sql = **"SELECT \* FROM Veiculo where Alugado = 'Sim'"** query = QSqlQuery(sql)  
  
 **return** query  
  
  
**Veiculo CTR**

**class** VeiculoCTR:  
 **def** CadastrarVeiculo(modelo, marca, anoModelo, placa, alugado, batido,  
 kmAtual, valorDiaria, descricao, tipoVeiculo):  
 veiculoDTO = VeiculoDTO  
 veiculoDTO.Modelo = modelo  
 veiculoDTO.Marca = marca  
 veiculoDTO.AnoModelo = anoModelo  
 veiculoDTO.Placa = placa  
 veiculoDTO.Alugado = alugado  
 veiculoDTO.Batido = batido  
 veiculoDTO.KmAtual = kmAtual  
 veiculoDTO.ValorDiaria = valorDiaria  
 veiculoDTO.Descricao = descricao  
 veiculoDTO.TipoVeiculo = tipoVeiculo  
  
 veiculoDAO = VeiculoDAO  
 veiculoDAO.CadastrarVeiculo(veiculoDTO)  
  
 **def** AtualizarVeiculo(codigoVeic, modelo, marca, anoModelo, placa, alugado, batido, kmAtual, valorDiaria, descricao, tipoVeiculo):  
 veiculoDTO = VeiculoDTO  
 veiculoDTO.Modelo = modelo  
 veiculoDTO.Marca = marca  
 veiculoDTO.AnoModelo = anoModelo  
 veiculoDTO.Placa = placa  
 veiculoDTO.Alugado = alugado  
 veiculoDTO.Batido = batido  
 veiculoDTO.KmAtual = kmAtual  
 veiculoDTO.ValorDiaria = valorDiaria  
 veiculoDTO.Descricao = descricao  
 veiculoDTO.TipoVeiculo = tipoVeiculo  
  
 veiculoDAO = VeiculoDAO  
 veiculoDAO.AtualizarVeiculo(codigoVeic, veiculoDTO)  
  
 **def** PesquisarVeiculosDisponiveis():  
 veiculoDAO = VeiculoDAO  
 query = veiculoDAO.PesquisarVeiculosDisponiveis()  
  
 **return** query  
  
 **def** PesquisarTodosVeiculos():  
 veiculoDAO = VeiculoDAO  
 query = veiculoDAO.PesquisarTodosVeiculos()  
  
 **return** query

**def** PesquisarVeiculo(valor, tipo):  
 veiculoDAO = VeiculoDAO  
 query = veiculoDAO.PesquisarVeiculo(valor, tipo)  
  
 **return** query  
  
 **def** ExcluirVeiculo(codigoVeic):  
 veiculoDAO = VeiculoDAO  
 veiculoDAO.ExcluirVeiculo(codigoVeic)

**Aluguel DTO**

**class** AluguelDTO:  
 **def** \_\_init\_\_(self):  
 self.DataAluguel = **None** self.DataPrazo = **None** self.DataDevolucao = **None** self.ValorAluguel = **None** self.ValorMulta = **None** self.KmEntrada = **None** self.KmSaida = **None** self.CodigoCli = **None** self.CodigoVeic = **None**

**Aluguel DAO**

**class** AluguelDAO:  
 **def** CadastrarAluguel(aluguel):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
  
  
 query.prepare(**"UPDATE Veiculo SET Alugado = 'Sim' WHERE CodigoVeic = "**+aluguel.CodigoVeic)  
 query.exec\_()  
 db.commit()  
  
 query.prepare(**"INSERT INTO Aluguel(DataAluguel, DataPrazo, DataDevolucao, ValorAluguel,ValorMulta, KmEntrada, KmSaida, CodigoCli, CodigoVeic) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)"**)  
 query.addBindValue(aluguel.DataAluguel)  
 query.addBindValue(aluguel.DataPrazo)  
 query.addBindValue(aluguel.DataDevolucao)  
 query.addBindValue(aluguel.ValorAluguel)  
 query.addBindValue(aluguel.ValorMulta)  
 query.addBindValue(aluguel.KmEntrada)  
 query.addBindValue(aluguel.KmSaida)  
 query.addBindValue(aluguel.CodigoCli)  
 query.addBindValue(aluguel.CodigoVeic)  
 query.exec\_()  
 db.commit()

**def** PesquisarTodosAluguel():  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 sql = **"SELECT Aluguel.\*, Cliente.Nome FROM Aluguel INNER JOIN Cliente ON Aluguel.CodigoCli = Cliente.CodigoCli"** query = QSqlQuery(sql)  
  
 **return** query  
  
 **def** PesquisarAluguel(valor, tipo):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 **if** tipo==**'Código Aluguel'**:  
 sql = **"SELECT Aluguel.\*, Cliente.Nome FROM Aluguel INNER JOIN Cliente ON Aluguel.CodigoCli = Cliente.CodigoCli where Aluguel.CodigoAlug = "** + valor  
 query = QSqlQuery(sql)  
 **elif** tipo==**'Código Cliente'**:  
 sql = **"SELECT Aluguel.\*, Cliente.Nome FROM Aluguel INNER JOIN Cliente ON Aluguel.CodigoCli = Cliente.CodigoCli where Aluguel.CodigoCli = "**+valor  
 query = QSqlQuery(sql)  
 **elif** tipo==**'Código Veículo'**:  
 sql = **"SELECT Aluguel.\*, Cliente.Nome FROM Aluguel INNER JOIN Cliente ON Aluguel.CodigoCli = Cliente.CodigoCli where Aluguel.CodigoVeic = "** + valor  
 query = QSqlQuery(sql)  
 **elif** tipo==**'Nome Cliente'**:  
 sql = **"SELECT Aluguel.\*, Cliente.Nome FROM Aluguel INNER JOIN Cliente ON Aluguel.CodigoCli = Cliente.CodigoCli where Cliente.Nome = '"** + valor+**"'"** query = QSqlQuery(sql)  
  
 **return** query  
  
 **def** DevolverVeiculo(codigoAlug, aluguel):  
 conn = ConexaoSQL  
 db = conn.getConexao()  
 db.open()  
  
 query = QSqlQuery()  
  
 query.prepare(**"UPDATE Veiculo SET Alugado = 'Não' WHERE CodigoVeic = "**+codigoVeic)  
 query.exec\_()  
 db.commit()  
  
 query.prepare(**"UPDATE Aluguel SET DataDevolucao = '"**+aluguel.DataDevolucao+**"', ValorMulta = '"**+aluguel.ValorMulta  
 +**"', KmSaida = '"**+aluguel.KmSaida  
 +**"' WHERE CodigoCli = "**+codigoAlug)  
 query.exec\_()  
 db.commit()

**Aluguel CTR**

**class** AluguelCTR:  
 **def** CadastrarAluguel(DataAluguel, DataPrazo, DataDevolucao, ValorAluguel,  
 ValorMulta, KmEntrada, KmSaida, CodigoCli, CodigoVeic):  
 aluguelDTO = AluguelDTO  
 aluguelDTO.DataAluguel = DataAluguel  
 aluguelDTO.DataPrazo = DataPrazo  
 aluguelDTO.DataDevolucao = DataDevolucao  
 aluguelDTO.ValorAluguel = ValorAluguel  
 aluguelDTO.ValorMulta = ValorMulta  
 aluguelDTO.KmEntrada = KmEntrada  
 aluguelDTO.KmSaida = KmSaida  
 aluguelDTO.CodigoCli = CodigoCli  
 aluguelDTO.CodigoVeic = CodigoVeic  
  
 aluguelDAO = AluguelDAO  
 aluguelDAO.CadastrarAluguel(aluguelDTO)  
  
 **def** PesquisarTodosAluguel():  
 aluguelDAO = AluguelDAO  
 query = aluguelDAO.PesquisarTodosAluguel()  
  
 **return** query  
  
 **def** PesquisarAluguel(valor, tipo):  
 aluguelDAO = AluguelDAO  
 query = aluguelDAO.PesquisarAluguel(valor, tipo)  
  
 **return** query  
  
 **def** DevolverVeiculo(codigoAlug, dataDevol, valorMulta, kmSaida):  
 aluguelDTO = AluguelDTO  
  
 aluguelDTO.DataDevolucao = dataDevol  
 aluguelDTO.ValorMulta = valorMulta  
 aluguelDTO.KmSaida = kmSaida  
  
  
 aluguelDAO = AluguelDAO  
 aluguelDAO.DevolverVeiculo(codigoAlug, aluguelDTO)

**APÊNDICE B – MÉTODOS DAS TELAS DE VIEW.**

**Cadastro de Cliente**

*#PREENCHER OS CAMPOS PARA ALTERAÇÃO***def** PreencherAlterar(self, nome, cpf, endereco, email, telefone):  
 self.edtNome.setText(nome)  
 self.edtCPF.setText(cpf)  
 self.edtEndereco.setText(endereco)  
 self.edtEmail.setText(email)  
 self.edtTelefone.setText(telefone)  
  
*#CLICK BTN\_SALVAR***def** btnSalvar\_Click(self, estado, codigoCli):  
 nome = self.edtNome.text()  
 cpf = self.edtCPF.text()  
 endereco = self.edtEndereco.text()  
 email = self.edtEmail.text()  
 telefone = self.edtTelefone.text()  
  
 *#VERIFICA O ESTADO INSERIR/ALTERAR PARA CHAMAR A FUNÇAO APROPRIADA* **if** estado==**'inserir'**:  
 cliente = ClienteCTR  
 cliente.CadastrarCliente(nome, cpf, endereco, email, telefone)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Cliente inserido com sucesso!"**)msg.setWindowTitle(**"Inserir Cliente"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
 **if** estado==**'alterar'**:  
 cliente = ClienteCTR  
 cliente.AtualizarCliente(codigoCli, nome, cpf, endereco, email, telefone)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Cliente alterado com sucesso!"**)msg.setWindowTitle(**"Alterar Cliente"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
 self.edtNome.setText(**''**)  
 self.edtCPF.setText(**''**)  
 self.edtEndereco.setText(**''**)  
 self.edtEmail.setText(**''**)  
 self.edtTelefone.setText(**''**)

**Listar Cliente**

**def** AlterarCliente\_Click(self):  
 linha = self.gridCliente.currentItem().row()  
 codigoCli = self.gridCliente.item(linha, 0).text()  
 nome = self.gridCliente.item(linha, 1).text()  
 cpf = self.gridCliente.item(linha, 2).text()  
 endereco = self.gridCliente.item(linha, 3).text()  
 email = self.gridCliente.item(linha, 4).text()  
 telefone = self.gridCliente.item(linha, 5).text()  
  
 self.frmCliente = QtGui.QMainWindow()  
 self.ui = Ui\_frmCliente()  
 self.ui.setupUi(self.frmCliente, **'alterar'**, codigoCli)  
 self.ui.PreencherAlterar(nome, cpf, endereco, email, telefone)  
 self.frmCliente.show()  
  
**def** ExcluirCliente\_Click(self):  
  
 linha = self.gridCliente.currentItem().row()  
 codigoCli = self.gridCliente.item(linha, 0).text()  
  
 self.gridCliente.removeRow(linha)  
 cliente = ClienteCTR  
 cliente.ExcluirCliente(codigoCli)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Cliente Excluído!"**) msg.setWindowTitle(**"Excluir Cliente"**) msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
  
**def** PesquisarCliente(self, valor, tipo):  
 **if** valor == **''**:  
 self.PesquisarTodosClientes()  
 **else**:  
 cliente = ClienteCTR  
 query = cliente.PesquisarCliente(valor, tipo)  
  
 **while** (self.gridCliente.rowCount() > 0):  
 self.gridCliente.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridCliente.insertRow(row)  
 codCli = QTableWidgetItem(str(query.value(0)))  
 nome = QTableWidgetItem(str(query.value(1)))  
 cpf = QTableWidgetItem(str(query.value(2)))  
 endereco = QTableWidgetItem(str(query.value(3)))  
 email = QTableWidgetItem(str(query.value(4)))  
 telefone = QTableWidgetItem(str(query.value(5)))  
  
 self.gridCliente.setItem(row, 0, codCli)  
 self.gridCliente.setItem(row, 1, nome)  
 self.gridCliente.setItem(row, 2, cpf)  
 self.gridCliente.setItem(row, 3, endereco)  
 self.gridCliente.setItem(row, 4, email)  
 self.gridCliente.setItem(row, 5, telefone)  
  
 row = row + 1  
  
 self.edtPesquisa.setText(**''**)

**Cadastro de Veículo**

*#PREENCHER OS CAMPOS PARA ALTERAÇÃO***def** PreencherAlterar(self, modelo, marca, anoModelo, placa, alugado, batido, kmAtual, valorDiaria, descricao, tipoVeiculo):  
 self.edtModelo.setText(modelo)  
 self.edtMarca.setText(marca)  
 self.edtAno.setText(anoModelo)  
 self.edtPlaca.setText(placa)  
 self.edtKm.setText(kmAtual)  
 self.edtDiaria.setText(valorDiaria)  
 self.txtDescricao.setPlainText(descricao)  
 self.edtTipo.setText(tipoVeiculo)  
  
 **if** alugado==**'Sim'**:  
 self.rbAlugado.setChecked(**True**)  
 self.rbDisponivel.setChecked(**False**)  
 **elif** alugado==**'Não'**:  
 self.rbDisponivel.setChecked(**True**)  
 self.rbAlugado.setChecked(**False**)  
  
 **if** batido==**'Sim'**:  
 self.rbBatido.setChecked(**True**)  
 self.rbPerfeito.setChecked(**False**)  
 **elif** batido==**'Não'**:  
 self.rbPerfeito.setChecked(**True**)  
 self.rbBatido.setChecked(**False**)  
  
  
*#CLICK BTN\_SALVAR***def** btnSalvar\_Click(self, estado, codigoVeic):  
 modelo = self.edtModelo.text()  
 marca = self.edtMarca.text()  
 anoModelo = self.edtAno.text()  
 placa = self.edtPlaca.text()  
  
 **if** self.rbAlugado.isChecked():  
 alugado = self.rbAlugado.text()  
 **elif** self.rbDisponivel.isChecked():  
 alugado = self.rbDisponivel.text()  
  
 **if** self.rbBatido.isChecked():  
 batido = self.rbBatido.text()  
 **elif** self.rbPerfeito.isChecked():  
 batido = self.rbPerfeito.text()  
  
 kmAtual = self.edtKm.text()  
 valorDiaria = self.edtDiaria.text()  
 descricao = self.txtDescricao.toPlainText()  
 tipoVeiculo = self.edtTipo.text()

*#VERIFICA O ESTADO INSERIR/ALTERAR PARA CHAMAR A FUNÇAO APROPRIADA* **if** estado==**'inserir'**:  
 veiculo = VeiculoCTR  
 veiculo.CadastrarVeiculo(modelo, marca, anoModelo, placa, alugado,  
 batido, kmAtual, valorDiaria, descricao, tipoVeiculo)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Veículo inserido com sucesso!"**)msg.setWindowTitle(**"Inserir Veículo"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
 **if** estado==**'alterar'**:  
 veiculo = VeiculoCTR  
 veiculo.AtualizarVeiculo(codigoVeic, modelo, marca, anoModelo, placa, alugado, batido, kmAtual, valorDiaria, descricao, tipoVeiculo)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Veículo alterado com sucesso!"**)msg.setWindowTitle(**"Alterar Veículo"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
 self.edtModelo.setText(**''**)  
 self.edtMarca.setText(**''**)  
 self.edtAno.setText(**''**)  
 self.edtPlaca.setText(**''**)  
 self.edtKm.setText(**''**)  
 self.edtDiaria.setText(**''**)  
 self.txtDescricao.setText(**''**)  
 self.edtTipo.setText(**''**)

**Listar Veículo**

**def** AlterarVeiculo\_Click(self):  
 linha = self.gridVeiculos.currentItem().row()  
 codigoVeic = self.gridVeiculos.item(linha, 0).text()  
 modelo = self.gridVeiculos.item(linha, 1).text()  
 marca = self.gridVeiculos.item(linha, 2).text()  
 anoModelo = self.gridVeiculos.item(linha, 3).text()  
 placa = self.gridVeiculos.item(linha, 4).text()  
 alugado = self.gridVeiculos.item(linha, 5).text()  
 batido = self.gridVeiculos.item(linha, 6).text()  
 kmAtual = self.gridVeiculos.item(linha, 7).text()  
 valorDiaria = self.gridVeiculos.item(linha, 8).text()  
 descricao = self.gridVeiculos.item(linha, 9).text()  
 tipoVeiculo = self.gridVeiculos.item(linha, 10).text()  
  
 self.frmVeiculos = QtGui.QMainWindow()  
 self.ui = Ui\_frmVeiculos()  
 self.ui.setupUi(self.frmVeiculos, **'alterar'**, codigoVeic)  
 self.ui.PreencherAlterar(modelo, marca, anoModelo, placa, alugado, batido, kmAtual, valorDiaria, descricao, tipoVeiculo)  
 self.frmVeiculos.show()  
**def** ExcluirVeiculo\_Click(self):  
  
 linha = self.gridVeiculos.currentItem().row()  
 codigoVeic = self.gridVeiculos.item(linha, 0).text()  
  
 self.gridVeiculos.removeRow(linha)  
 veiculo = VeiculoCTR  
 veiculo.ExcluirVeiculo(codigoVeic)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Veículo Excluído!"**)msg.setWindowTitle(**"Excluir Veículo"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
  
**def** PesquisarVeiculo(self, valor, tipo):  
 **if** (valor == **''**) **and** (tipo!=**'Disponível'**) **and** (tipo!=**'Alugado'**):  
 self.PesquisarTodosVeiculos()  
 **else**:  
 veiculo = VeiculoCTR  
 query = veiculo.PesquisarVeiculo(valor, tipo)  
  
 **while** (self.gridVeiculos.rowCount() > 0):  
 self.gridVeiculos.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridVeiculos.insertRow(row)  
 codigoVeic = QTableWidgetItem(str(query.value(0)))  
 modelo = QTableWidgetItem(str(query.value(1)))  
 marca = QTableWidgetItem(str(query.value(2)))  
 anoModelo = QTableWidgetItem(str(query.value(3)))  
 placa = QTableWidgetItem(str(query.value(4)))  
 alugado = QTableWidgetItem(str(query.value(5)))  
 batido = QTableWidgetItem(str(query.value(6)))  
 kmAtual = QTableWidgetItem(str(query.value(7)))  
 valorDiaria = QTableWidgetItem(str(query.value(8)))  
 descricao = QTableWidgetItem(str(query.value(9)))  
 tipoVeiculo = QTableWidgetItem(str(query.value(10)))  
  
 self.gridVeiculos.setItem(row, 0, codigoVeic)  
 self.gridVeiculos.setItem(row, 1, modelo)  
 self.gridVeiculos.setItem(row, 2, marca)  
 self.gridVeiculos.setItem(row, 3, anoModelo)  
 self.gridVeiculos.setItem(row, 4, placa)  
 self.gridVeiculos.setItem(row, 5, alugado)  
 self.gridVeiculos.setItem(row, 6, batido)  
 self.gridVeiculos.setItem(row, 7, kmAtual)  
 self.gridVeiculos.setItem(row, 8, valorDiaria)  
 self.gridVeiculos.setItem(row, 9, descricao)  
 self.gridVeiculos.setItem(row, 10, tipoVeiculo)  
  
 row = row + 1  
  
 self.edtPesquisa.setText(**''**)

**def** PesquisarTodosVeiculos(self):  
 veiculo = VeiculoCTR  
 query = veiculo.PesquisarTodosVeiculos()  
  
 **while** (self.gridVeiculos.rowCount() > 0):  
 self.gridVeiculos.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridVeiculos.insertRow(row)  
 codigoVeic = QTableWidgetItem(str(query.value(0)))  
 modelo = QTableWidgetItem(str(query.value(1)))  
 marca = QTableWidgetItem(str(query.value(2)))  
 anoModelo = QTableWidgetItem(str(query.value(3)))  
 placa = QTableWidgetItem(str(query.value(4)))  
 alugado = QTableWidgetItem(str(query.value(5)))  
 batido = QTableWidgetItem(str(query.value(6)))  
 kmAtual = QTableWidgetItem(str(query.value(7)))  
 valorDiaria = QTableWidgetItem(str(query.value(8)))  
 descricao = QTableWidgetItem(str(query.value(9)))  
 tipoVeiculo = QTableWidgetItem(str(query.value(10)))  
  
 self.gridVeiculos.setItem(row, 0, codigoVeic)  
 self.gridVeiculos.setItem(row, 1, modelo)  
 self.gridVeiculos.setItem(row, 2, marca)  
 self.gridVeiculos.setItem(row, 3, anoModelo)  
 self.gridVeiculos.setItem(row, 4, placa)  
 self.gridVeiculos.setItem(row, 5, alugado)  
 self.gridVeiculos.setItem(row, 6, batido)  
 self.gridVeiculos.setItem(row, 7, kmAtual)  
 self.gridVeiculos.setItem(row, 8, valorDiaria)  
 self.gridVeiculos.setItem(row, 9, descricao)  
 self.gridVeiculos.setItem(row, 10, tipoVeiculo)  
  
 row = row + 1

**Cadastrar Aluguel**

*#CLICK BTN\_SALVAR***def** btnSalvar\_Click(self):  
 linha = self.gridCliente.currentItem().row()  
 codigoCli = self.gridCliente.item(linha, 0).text()  
 linha = self.gridVeiculo.currentItem().row()  
 codigoVeic = self.gridVeiculo.item(linha, 0).text()  
  
 DataAluguel = self.EdtDataAluguel.text()  
 DataPrazo = self.EdtPrazo.text()  
 DataDevolucao = self.EdtDataDev.text()  
 ValorAluguel = self.EdtValor.text()  
 ValorMulta = self.EdtMulta.text()  
 KmEntrada = self.EdtkmEntrada.text()  
 KmSaida = self.EdtKmSaida.text()  
  
 aluguel = AluguelCTR  
 aluguel.CadastrarAluguel(DataAluguel, DataPrazo, DataDevolucao, ValorAluguel, ValorMulta, KmEntrada, KmSaida, codigoCli, codigoVeic)

msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Aluguel cadastrado com sucesso!"**)  
 msg.setWindowTitle(**"Cadastro de Aluguel"**)  
 msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
 self.EdtDataAluguel.setText(**''**)  
 self.EdtPrazo.setText(**''**)  
 self.EdtDataDev.setText(**''**)  
 self.EdtValor.setText(**''**)  
 self.EdtMulta.setText(**''**)  
 self.EdtkmEntrada.setText(**''**)  
 self.EdtKmSaida.setText(**''**)  
  
  
**def** PesquisarTodosClientes(self):  
 cliente = ClienteCTR  
 query = cliente.PesquisarTodosClientes()  
  
 **while** (self.gridCliente.rowCount() > 0):  
 self.gridCliente.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridCliente.insertRow(row)  
 codCli = QTableWidgetItem(str(query.value(0)))  
 nome = QTableWidgetItem(str(query.value(1)))  
 cpf = QTableWidgetItem(str(query.value(2)))  
 endereco = QTableWidgetItem(str(query.value(3)))  
 email = QTableWidgetItem(str(query.value(4)))  
 telefone = QTableWidgetItem(str(query.value(5)))  
  
 self.gridCliente.setItem(row, 0, codCli)  
 self.gridCliente.setItem(row, 1, nome)  
 self.gridCliente.setItem(row, 2, cpf)  
 self.gridCliente.setItem(row, 3, endereco)  
 self.gridCliente.setItem(row, 4, email)  
 self.gridCliente.setItem(row, 5, telefone)  
  
 row = row + 1  
  
**def** PesquisarCliente(self, valor, tipo):  
 **if** valor == **''**:  
 self.PesquisarTodosClientes()  
 **else**:  
 cliente = ClienteCTR  
 query = cliente.PesquisarCliente(valor, tipo)  
  
 **while** (self.gridCliente.rowCount() > 0):  
 self.gridCliente.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridCliente.insertRow(row)  
 codCli = QTableWidgetItem(str(query.value(0)))  
 nome = QTableWidgetItem(str(query.value(1)))  
 cpf = QTableWidgetItem(str(query.value(2)))  
 endereco = QTableWidgetItem(str(query.value(3)))  
 email = QTableWidgetItem(str(query.value(4)))  
 telefone = QTableWidgetItem(str(query.value(5)))  
  
 self.gridCliente.setItem(row, 0, codCli)  
 self.gridCliente.setItem(row, 1, nome)  
 self.gridCliente.setItem(row, 2, cpf)  
 self.gridCliente.setItem(row, 3, endereco)  
 self.gridCliente.setItem(row, 4, email)  
 self.gridCliente.setItem(row, 5, telefone)  
  
 row = row + 1  
  
**def** PesquisarVeiculo(self, valor, tipo):  
 **if** (valor == **''**) :  
 self.PesquisarVeiculosDisponiveis()  
 **else**:  
 veiculo = VeiculoCTR  
 query = veiculo.PesquisarVeiculo(valor, tipo)  
  
 **while** (self.gridVeiculo.rowCount() > 0):  
 self.gridVeiculo.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridVeiculo.insertRow(row)  
 codigoVeic = QTableWidgetItem(str(query.value(0)))  
 modelo = QTableWidgetItem(str(query.value(1)))  
 marca = QTableWidgetItem(str(query.value(2)))  
 anoModelo = QTableWidgetItem(str(query.value(3)))  
 placa = QTableWidgetItem(str(query.value(4)))  
 alugado = QTableWidgetItem(str(query.value(5)))  
 batido = QTableWidgetItem(str(query.value(6)))  
 kmAtual = QTableWidgetItem(str(query.value(7)))  
 valorDiaria = QTableWidgetItem(str(query.value(8)))  
 descricao = QTableWidgetItem(str(query.value(9)))  
 tipoVeiculo = QTableWidgetItem(str(query.value(10)))  
  
 self.gridVeiculo.setItem(row, 0, codigoVeic)  
 self.gridVeiculo.setItem(row, 1, modelo)  
 self.gridVeiculo.setItem(row, 2, marca)  
 self.gridVeiculo.setItem(row, 3, anoModelo)  
 self.gridVeiculo.setItem(row, 4, placa)  
 self.gridVeiculo.setItem(row, 5, alugado)  
 self.gridVeiculo.setItem(row, 6, batido)  
 self.gridVeiculo.setItem(row, 7, kmAtual)  
 self.gridVeiculo.setItem(row, 8, valorDiaria)  
 self.gridVeiculo.setItem(row, 9, descricao)  
 self.gridVeiculo.setItem(row, 10, tipoVeiculo)  
  
 row = row + 1  
  
**def** PesquisarVeiculosDisponiveis(self):  
 veiculo = VeiculoCTR  
 query = veiculo.PesquisarVeiculosDisponiveis()  
  
 **while** (self.gridVeiculo.rowCount() > 0):  
 self.gridVeiculo.removeRow(0)  
  
 row = 0

**while** query.next():  
 self.gridVeiculo.insertRow(row)  
 codigoVeic = QTableWidgetItem(str(query.value(0)))  
 modelo = QTableWidgetItem(str(query.value(1)))  
 marca = QTableWidgetItem(str(query.value(2)))  
 anoModelo = QTableWidgetItem(str(query.value(3)))  
 placa = QTableWidgetItem(str(query.value(4)))  
 alugado = QTableWidgetItem(str(query.value(5)))  
 batido = QTableWidgetItem(str(query.value(6)))  
 kmAtual = QTableWidgetItem(str(query.value(7)))  
 valorDiaria = QTableWidgetItem(str(query.value(8)))  
 descricao = QTableWidgetItem(str(query.value(9)))  
 tipoVeiculo = QTableWidgetItem(str(query.value(10)))  
  
 self.gridVeiculo.setItem(row, 0, codigoVeic)  
 self.gridVeiculo.setItem(row, 1, modelo)  
 self.gridVeiculo.setItem(row, 2, marca)  
 self.gridVeiculo.setItem(row, 3, anoModelo)  
 self.gridVeiculo.setItem(row, 4, placa)  
 self.gridVeiculo.setItem(row, 5, alugado)  
 self.gridVeiculo.setItem(row, 6, batido)  
 self.gridVeiculo.setItem(row, 7, kmAtual)  
 self.gridVeiculo.setItem(row, 8, valorDiaria)  
 self.gridVeiculo.setItem(row, 9, descricao)  
 self.gridVeiculo.setItem(row, 10, tipoVeiculo)  
  
 row = row + 1

**Listar Aluguel**

**def** DevolverVeiculo(self):  
 linha = self.gridAluguel.currentItem().row()  
 codigoAlug = self.gridAluguel.item(linha, 0).text()  
 dataDevol = self.edtDevolucao.text()  
 valorMulta = self.edtMulta.text()  
 kmSaida = self.edtSaida.text()  
  
 aluguelCTR = AluguelCTR  
 aluguelCTR.DevolverVeiculo(codigoAlug, dataDevol, valorMulta, kmSaida)  
  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText(**"Veículo devolvido!"**)msg.setWindowTitle(**"Devolver Veículo"**)msg.setStandardButtons(QMessageBox.Ok)  
 msg.exec\_()  
  
 self.edtDevolucao.setText(**''**)  
 self.edtMulta.setText(**''**)  
 self.edtSaida.setText(**''**)  
  
**def** PesquisarAluguel(self, valor, tipo):  
 **if** (valor == **''**):  
 self.PesquisarTodosAluguel()  
 **else**:  
 aluguel = AluguelCTR  
 query = aluguel.PesquisarAluguel(valor, tipo)  
 **while** (self.gridAluguel.rowCount() > 0):  
 self.gridAluguel.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridAluguel.insertRow(row)  
 codigoAlug = QTableWidgetItem(str(query.value(0)))  
 nomecliente = QTableWidgetItem(str(query.value(10)))  
 dataAlug = QTableWidgetItem(str(query.value(1)))  
 dataPrazo = QTableWidgetItem(str(query.value(2)))  
 dataDevolucao = QTableWidgetItem(str(query.value(3)))  
 valorAluguel = QTableWidgetItem(str(query.value(4)))  
 valorMulta = QTableWidgetItem(str(query.value(5)))  
 kmEntrada = QTableWidgetItem(str(query.value(6)))  
 kmSaida = QTableWidgetItem(str(query.value(7)))  
  
 self.gridAluguel.setItem(row, 0, codigoAlug)  
 self.gridAluguel.setItem(row, 1, nomecliente)  
 self.gridAluguel.setItem(row, 2, dataAlug)  
 self.gridAluguel.setItem(row, 3, dataPrazo)  
 self.gridAluguel.setItem(row, 4, dataDevolucao)  
 self.gridAluguel.setItem(row, 5, valorAluguel)  
 self.gridAluguel.setItem(row, 6, valorMulta)  
 self.gridAluguel.setItem(row, 7, kmEntrada)  
 self.gridAluguel.setItem(row, 8, kmSaida)  
  
 row = row + 1  
  
 self.edtPesquisa.setText(**''**)  
  
**def** PesquisarTodosAluguel(self):  
 aluguel = AluguelCTR  
 query = aluguel.PesquisarTodosAluguel()  
  
 **while** (self.gridAluguel.rowCount() > 0):  
 self.gridAluguel.removeRow(0)  
  
 row = 0  
 **while** query.next():  
 self.gridAluguel.insertRow(row)  
 codigoAlug = QTableWidgetItem(str(query.value(0)))  
 nomecliente = QTableWidgetItem(str(query.value(10)))  
 dataAlug = QTableWidgetItem(str(query.value(1)))  
 dataPrazo = QTableWidgetItem(str(query.value(2)))  
 dataDevolucao = QTableWidgetItem(str(query.value(3)))  
 valorAluguel = QTableWidgetItem(str(query.value(4)))  
 valorMulta = QTableWidgetItem(str(query.value(5)))  
 kmEntrada = QTableWidgetItem(str(query.value(6)))  
 kmSaida = QTableWidgetItem(str(query.value(7)))  
  
 self.gridAluguel.setItem(row, 0, codigoAlug)  
 self.gridAluguel.setItem(row, 1, nomecliente)  
 self.gridAluguel.setItem(row, 2, dataAlug)  
 self.gridAluguel.setItem(row, 3, dataPrazo)  
 self.gridAluguel.setItem(row, 4, dataDevolucao)  
 self.gridAluguel.setItem(row, 5, valorAluguel)  
 self.gridAluguel.setItem(row, 6, valorMulta)  
 self.gridAluguel.setItem(row, 7, kmEntrada)  
 self.gridAluguel.setItem(row, 8, kmSaida)  
  
 row = row + 1