self.frameDadosCliente.hide()  
  
 self.lb\_observacao.setFocus() # *Todo: faz funciona o texto da area grande* # *TODO: Botões sair do sistema* self.bt\_sair.clicked.connect(formCliente.close) # *Todo: sai da tela.* self.bt\_excluir.clicked.connect(self.excluirCliente) # *Todo: exclui o cliente.* self.bt\_pesquisarGeral.clicked.connect(self.consultarGeral) # *Todo: pesquisa Geral* self.bt\_pesquisar.clicked.connect(self.consultarNomes) # *Todo: consulta por nome* self.bt\_retornar.clicked.connect(self.retornarPagTab) # *Todo: Retorna a pagina 0* self.bt\_avancar.clicked.connect(self.avancaPagTab) # *Todo: vai para a página 1* self.bt\_confirmar.clicked.connect(self.cadastrarCliente) # *Todo: Cadastra cliente no banco de dados.* self.bt\_alterar.clicked.connect(self.abrirDadosClientes)  
  
 self.bt\_consultar.clicked.connect(self.consultarCliente)  
  
 self.bt\_alterarDados.clicked.connect(self.alterarDadosClientes)  
  
 self.bt\_incluir.clicked.connect(self.incluirNovoCliente)  
  
 self.consultarGeral()  
  
  
  
  
 # *TODO: funções da tela cliente.*def sairTelaCliente(self):  
 sys.exit()  
  
def retornarPagTab(self):  
 self.tabWidget.setCurrentIndex(0)  
  
def avancaPagTab(self):  
 self.tabWidget.setCurrentIndex(1)  
  
def cadastrarCliente(self):  
  
  
 self.limpaTela()  
 self.tabWidget.setCurrentIndex(0)  
  
 self.lb\_razaoSocial.setText('')  
 # *Todo: seta labels das variaveis.* razaoSocial = self.lb\_razaoSocial.text() # 1  
 nomeFantasia = self.lb\_nomeFantasia.text() # 2  
 classificacao = self.lb\_class.text() # 3  
 cnpjcpf = self.lb\_cnpjcpf.text() # 4  
 ierg = self.lb\_ierg.text() # 5  
 endereco = self.lb\_endereco.text() # 6  
 numero = self.lb\_numero.text() # 7  
 bairro = self.lb\_bairro.text() # 8  
 cidade = self.lb\_cidade.text() # 9  
 cep = self.lb\_cep.text() # 10  
 estado = self.lb\_uf.text() # 11  
 telefone = self.lb\_telefone.text() # 12  
 email = self.lb\_email.text() # 13  
 observacao = self.lb\_observacao.toPlainText() # 14  
 prazo = self.lb\_prazo.text() # 15  
 usuarioCadastro = 1 # 16  
 dataHoje = datetime.today().strftime('%H:%M:%S - %d-%m-%Y')  
 serialUniqPC = subprocess.check\_output('wmic bios get serialnumber').decode("utf-8")  
 auditoria = "Hora Exata:" + str(dataHoje) + ' Máquina: ' + str(  
 serialUniqPC[36:49] + ' Usuário: ' + str(usuarioCadastro)) # 17  
  
 print(dataHoje)  
 print('cadastrado com sucesso')  
  
 # *todo: testando as captações* """print(razaoSocial)  
 print(nomeFantasia)  
 print(classificacao)  
 print(cnpjcpf)  
 print(ierg)  
 print(endereco)  
 print(numero)  
 print(bairro)  
 print(cidade)  
 print(cep)  
 print(estado)  
 print(telefone)  
 print(email)  
 print(observacao)  
 print(prazo)  
 print(auditoria)"""  
  
 # *Todo: fazer uma checagem se esta tudo vazio com a finalidade de não haver erros* if razaoSocial == '':  
 razaoaSocial = None  
 if nomeFantasia == '':  
 nomeFantasia = None  
 if classificacao == '':  
 classificacao = None  
 if cnpjcpf == '':  
 cnpjcpf = None  
 if ierg == '':  
 ierg = None  
 if endereco == '':  
 endereco = None  
 if numero == '':  
 numero = None  
 if bairro == '':  
 bairro = None  
 if cidade == '':  
 cidade = None  
 if cep == '':  
 cep = None  
 if estado == '':  
 estado = None  
 if telefone == '':  
 telefone = None  
 if email == '':  
 email = None  
 if observacao == '':  
 observacao = None  
 if prazo == '':  
 prazo = None  
  
  
 try:  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
 sql = "INSERT INTO cliente (razaoSocial, nomeFantasia, classificacao, cnpjcpf, ierg, endereco, numero, bairro, cidade, cep, estado, telefone, email, observacao, prazo, usuarioCadastro, auditoria) VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"  
 val = (  
 razaoSocial, nomeFantasia, classificacao, cnpjcpf, ierg, endereco, numero, bairro, cidade, cep, estado,  
 telefone, email, observacao, prazo, usuarioCadastro, auditoria)  
 mycursor.execute(sql, val)  
 # M commit o banco de dados  
 mydb.commit()  
  
 self.lb\_razaoSocial.clear()  
 self.lb\_nomeFantasia.clear()  
 self.lb\_class.clear()  
 self.lb\_cnpjcpf.clear()  
 self.lb\_ierg.clear()  
 self.lb\_endereco.clear()  
 self.lb\_numero.clear()  
 self.lb\_bairro.clear()  
 self.lb\_cidade.clear()  
 self.lb\_cep.clear()  
 self.lb\_uf.clear()  
 self.lb\_telefone.clear()  
 self.lb\_email.clear()  
 self.lb\_observacao.clear()  
 self.lb\_prazo.clear()  
 self.tabWidget.setCurrentIndex(0) # *Todo: volta página 0* msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setWindowTitle("Adição")  
 msg.setText(f" Adicionado com sucesso!")  
 msg.setStyleSheet("color: rgb(0, 170, 0)")  
 msg.exec\_()  
  
  
  
  
  
  
  
 except:  
 print('')  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.critical)  
 msg.setText("Erro SQL")  
 msg.setInformativeText('Favor contactar o administrador do sistema!')  
 msg.setWindowTitle('Erro')  
 msg.exec\_()  
  
 mycursor.close()  
 mydb.close()  
  
#*Todo: alterar*def abrirDadosClientes(self):  
 self.consultarGeral()  
 if self.tb\_cliente.rowCount()>1:  
  
  
  
  
 print()  
 linha = self.tb\_cliente.currentRow()  
 idClienteLinha = self.tb\_cliente.item(linha, 0).text()  
 print(f"Linha: {linha}, ID Cliente: {idClienteLinha}")  
  
  
  
 try:  
 mydb = mysql.connector.connect(  
 host = host,  
 user = user,  
 password = password,  
 database = database  
  
 )  
 mycursor = mydb.cursor()  
 consultaSQL = f"SELECT \* FROM cliente WHERE idCliente = " + idClienteLinha  
 mycursor.execute(consultaSQL)  
 myresult = mycursor.fetchall()  
 print(myresult)  
  
  
  
 df\_cliente = pd.DataFrame(myresult,  
 columns=['idCliente', 'razaoSocial', 'nomeFantasia', 'classificacao', 'cnpjcpf',  
 'ierg', 'endereco', 'numero', 'bairro', 'cep', 'cidade', 'estado',  
 'telefone', 'email', 'observacao', 'prazo', 'usuarioCadastro',  
 'auditoria'])  
  
 self.lb\_razaoSocial.setText(df\_cliente.loc[0][1])  
 self.lb\_nomeFantasia.setText(df\_cliente.loc[0][2])  
 self.lb\_class.setText(df\_cliente.loc[0][3])  
 self.lb\_cnpjcpf.setText(df\_cliente.loc[0][4])  
 self.lb\_ierg.setText(df\_cliente.loc[0][5])  
 self.lb\_endereco.setText(df\_cliente.loc[0][6])  
  
  
  
 numeroCasa = int(df\_cliente.loc[0][7])  
 self.lb\_numero.setText(str(numeroCasa))  
  
 self.lb\_bairro.setText(df\_cliente.loc[0][8])  
 self.lb\_cep.setText(df\_cliente.loc[0][9])  
 self.lb\_uf.setText(df\_cliente.loc[0][10])  
 self.lb\_cidade.setText(df\_cliente.loc[0][11])  
 self.lb\_telefone.setText(df\_cliente.loc[0][12])  
 self.lb\_email.setText(df\_cliente.loc[0][13])  
 self.lb\_observacao.insertPlainText(df\_cliente.loc[0][14])  
 self.lb\_prazo.setText(df\_cliente.loc[0][15])  
  
  
  
 except mysql.connector.Error:  
 print('Erro')  
 else:  
 print('Carrega a Tabela')  
 self.consultarGeral()  
 self.tb\_cliente.selectRow(0)  
  
  
def alterarDadosClientes(self):  
 self.consultarGeral()  
 self.tb\_cliente.selectRow(0)  
  
  
 if self.tb\_cliente.rowCount() > 1:  
  
  
  
 print('Alterando')  
 linha = self.tb\_cliente.currentRow()  
 idClienteLinha = self.tb\_cliente.item(linha, 0).text()  
 print(f"Linha: {linha}, ID Cliente: {idClienteLinha}")  
 # *Todo: seta labels das variaveis.* razaoSocial = self.lb\_razaoSocial.text() # 1  
 nomeFantasia = self.lb\_nomeFantasia.text() # 2  
 classificacao = self.lb\_class.text() # 3  
 cnpjcpf = self.lb\_cnpjcpf.text() # 4  
 ierg = self.lb\_ierg.text() # 5  
 endereco = self.lb\_endereco.text() # 6  
 numero = self.lb\_numero.text() # 7  
 bairro = self.lb\_bairro.text() # 8  
 cidade = self.lb\_cidade.text() # 9  
 cep = self.lb\_cep.text() # 10  
 estado = self.lb\_uf.text() # 11  
 telefone = self.lb\_telefone.text() # 12  
 email = self.lb\_email.text() # 13  
 observacao = self.lb\_observacao.toPlainText() # 14  
 prazo = self.lb\_prazo.text() # 15  
 usuarioCadastro = 1 # 16  
 dataHoje = datetime.today().strftime('%H:%M:%S - %d-%m-%Y')  
 serialUniqPC = subprocess.check\_output('wmic bios get serialnumber').decode("utf-8")  
 auditoria = "Hora Exata:" + str(dataHoje) + ' Máquina: ' + str(  
 serialUniqPC[36:49] + ' Usuário: ' + str(usuarioCadastro)) # 17  
  
 print(dataHoje)  
 print('Alterado com sucesso')  
  
 # *todo: testando as captações* print(razaoSocial)  
 print(nomeFantasia)  
 print(classificacao)  
 print(cnpjcpf)  
 print(ierg)  
 print(endereco)  
 print(numero)  
 print(bairro)  
 print(cidade)  
 print(cep)  
 print(estado)  
 print(telefone)  
 print(email)  
 print(observacao)  
 print(prazo)  
 print(auditoria)  
  
 # *Todo: fazer uma checagem se esta tudo vazio com a finalidade de não haver erros* if razaoSocial == '':  
 razaoaSocial = None  
 if nomeFantasia == '':  
 nomeFantasia = None  
 if classificacao == '':  
 classificacao = None  
 if cnpjcpf == '':  
 cnpjcpf = None  
 if ierg == '':  
 ierg = None  
 if endereco == '':  
 endereco = None  
 if numero == '':  
 numero = None  
 if bairro == '':  
 bairro = None  
 if cidade == '':  
 cidade = None  
 if cep == '':  
 cep = None  
 if estado == '':  
 estado = None  
 if telefone == '':  
 telefone = None  
 if email == '':  
 email = None  
 if observacao == '':  
 observacao = None  
 if prazo == '':  
 prazo = None  
  
 try:  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
  
 sql = f"UPDATE cliente SET razaoSocial = '{razaoSocial}', nomeFantasia = '{nomeFantasia}', classificacao = '{classificacao}', cnpjcpf = '{cnpjcpf}', ierg = '{ierg}', endereco = '{endereco}', numero = '{numero}', bairro = '{bairro}', cep = '{cep}', cidade = '{cidade}', estado = '{estado}', telefone = '{telefone}', email = '{email}', observacao = '{observacao}', prazo = '{prazo}' WHERE idCliente = '{idClienteLinha}'"  
  
 mycursor.execute(sql)  
 mydb.commit()  
 self.lb\_razaoSocial.clear()  
 self.lb\_nomeFantasia.clear()  
 self.lb\_class.clear()  
 self.lb\_cnpjcpf.clear()  
 self.lb\_ierg.clear()  
 self.lb\_endereco.clear()  
 self.lb\_numero.clear()  
 self.lb\_bairro.clear()  
 self.lb\_cidade.clear()  
 self.lb\_cep.clear()  
 self.lb\_uf.clear()  
 self.lb\_telefone.clear()  
 self.lb\_email.clear()  
 self.lb\_observacao.clear()  
 self.lb\_prazo.clear()  
  
 mydb.close()  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setWindowTitle('alteração')  
 msg.setText(f'ID: {idClienteLinha} alterado com sucesso')  
 msg.exec()  
 self.limpaTela()  
 self.tabWidget.setCurrentIndex(0) # *Todo: volta página* print(f'\033[0;34m{idClienteLinha}\033[m')  
 print('Id de alteração: ', idClienteLinha)  
 except:  
 print('')  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Critical)  
 msg.setText('Erro SQL')  
 msg.setInformativeText('Favor contactar o Administrador do sistema')  
 msg.setWindowTitle('Erro')  
 msg.exec\_()  
 else:  
 self.consultarGeral()  
  
  
def consultarCliente(self):  
 self.consultarGeral()  
 self.abrirDadosClientes()  
  
  
 if self.tb\_cliente.rowCount() > 1:  
  
  
  
  
 #self.lb\_razaoSocial.setStyleSheet('color: rgb(0,0,250); background-color: rgb(229, 229, 229);')  
 linha = self.tb\_cliente.currentRow()  
 idConsultaCliente = self.tb\_cliente.item(linha, 0).text()  
  
 print(idConsultaCliente)  
 try:  
 mydb = mysql.connector.connect(  
 host = host,  
 user = user,  
 password = password,  
 database = database  
  
 )  
  
 mycursor = mydb.cursor()  
 consultaUsuario = self.lb\_nomeCliente.text()  
 consultaSQL = f"SELECT \* FROM cliente WHERE idCliente = '{idConsultaCliente}' ORDER BY razaoSocial"  
 mycursor.execute(consultaSQL)  
 myresult = mycursor.fetchall()  
 print(myresult)  
 df\_cliente = pd.DataFrame(myresult,columns=['idCliente', 'razaoSocial', 'nomeFantasia', 'classificacao', 'cnpjcpf',  
 'ierg', 'endereco', 'numero', 'bairro', 'cep', 'cidade', 'estado',  
 'telefone', 'email', 'observacao', 'prazo', 'usuarioCadastro',  
 'auditoria'])  
  
 razaoSocial = df\_cliente['idCliente'][0]  
 self.lb\_razaoSocial.setText(razaoSocial)  
 self.lb\_nomeFantasia.setText(df\_cliente.loc[0][2])  
 self.lb\_class.setText(df\_cliente.loc[0][3])  
 self.lb\_cnpjcpf.setText(df\_cliente.loc[0][4])  
 self.lb\_ierg.setText(df\_cliente.loc[0][5])  
 self.lb\_endereco.setText(df\_cliente.loc[0][6])  
  
  
  
 numeroCasa = int(df\_cliente.loc[0][7])  
 self.lb\_numero.setText(str(numeroCasa))  
  
 self.lb\_bairro.setText(df\_cliente.loc[0][8])  
 self.lb\_cep.setText(df\_cliente.loc[0][9])  
 self.lb\_uf.setText(df\_cliente.loc[0][10])  
 self.lb\_cidade.setText(df\_cliente.loc[0][11])  
 self.lb\_telefone.setText(df\_cliente.loc[0][12])  
 self.lb\_email.setText(df\_cliente.loc[0][13])  
 self.lb\_observacao.insertPlainText(df\_cliente.loc[0][14])  
 self.lb\_prazo.setText(df\_cliente.loc[0][15])  
  
 except:  
 print('Erro')  
 else:  
 self.consultarGeral()  
 self.tb\_cliente.selectRow(0)  
  
  
  
  
  
def excluirCliente(self):  
  
 if self.tb\_cliente.rowCount()>1:  
  
  
  
  
  
 print('Excluindo cliente....')  
 msg = QMessageBox()  
 msg.setWindowTitle("Aviso")  
 msg.setText("Deseja realmente Excluir os dados")  
 msg.setStandardButtons(QMessageBox.StandardButton.Yes | QMessageBox.StandardButton.No)  
 ret = msg.exec()  
  
 if ret == QMessageBox.StandardButton.Yes:  
 linha = self.tb\_cliente.currentRow()  
 idClienteLinha = self.tb\_cliente.item(linha, 0).text()  
  
 try:  
  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
 sql = f"DELETE From cliente WHERE idCliente = '{idClienteLinha}'"  
 mycursor.execute(sql)  
 mydb.commit()  
 #print(mycursor.rowcount, "Excluído com sucesso")  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setWindowTitle("Aviso")  
 msg.setText(f"Cliente {idClienteLinha} Excluido com sucesso")  
 msg.exec\_()  
 self.consultarGeral()  
  
  
 except:  
  
 self.all\_data = 0  
 print('tente depois')  
 msg = QMessageBox()  
 msg.setWindowTitle("Aviso")  
 msg.setText('Não foi possível fazer a exclusão')  
 msg.exec\_()  
  
 msg.setStandardButtons(QMessageBox.StandardButton.Ok)  
 ret = msg.exec()  
 else:  
 print()  
 #msg = QMessageBox()  
 #msg.setWindowTitle("Aviso")  
 #msg.setText("Cliente não excluido")  
 #msg.setStandardButtons(QMessageBox.StandardButton.Ok)  
 #ret = msg.exec()  
  
 try:  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
 sql = "SELECT \* FROM cliente"  
 mycursor.execute(sql)  
 myresult = mycursor.fetchall()  
  
 except:  
 self.all\_data = 0  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.warning)  
 msg.setText("Error")  
 msg.setInformativeText("Não foi possível consultar a tabela!")  
 msg.setWindowTitle("Error")  
 msg.exec\_()  
 pass  
 else:  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.Information)  
 msg.setText("TABELA VAZIA")  
 msg.setInformativeText("Carregaremos a tabela antes.")  
 msg.setWindowTitle("Error")  
 msg.exec\_()  
 pass  
 self.consultarGeral()  
  
  
  
def consultarGeral(self):  
  
  
  
 try:  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
 sql = "SELECT \* FROM cliente"  
 mycursor.execute(sql)  
 myresult = mycursor.fetchall()  
  
 df\_cliente = pd.DataFrame(myresult,  
 columns=['idCliente', 'razaoSocial', 'nomeFantasia', 'classificacao', 'cnpjcpf',  
 'ierg', 'endereco', 'numero', 'bairro', 'cep', 'cidade', 'estado',  
 'telefone', 'email', 'observacao', 'prazo', 'usuarioCadastro',  
 'auditoria'])  
 self.all\_data = df\_cliente  
 # *Todo: Carrega o arquivo na Tabela cliente* numRows = len(self.all\_data.index)  
 self.tb\_cliente.setColumnCount(len(self.all\_data.columns))  
 self.tb\_cliente.setRowCount(numRows)  
 # self.tb\_cliente.setHorizontalHeaderLabels(self.all\_data.columns) #Setar o título e cabeçalho da tabela  
  
 for i in range(numRows):  
 for j in range(len(self.all\_data.columns)):  
 self.tb\_cliente.setItem(i, j, QTableWidgetItem(str(self.all\_data.iat[i, j])))  
  
 self.tb\_cliente.resizeColumnsToContents()  
 self.tb\_cliente.resizeRowsToContents()  
  
 mycursor.close()  
  
  
  
  
  
  
 except:  
 self.all\_data = 0  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.warning)  
 msg.setText("Error")  
 msg.setInformativeText("Não foi possível consultar a tabela!")  
 msg.setWindowTitle("Error")  
 msg.exec\_()  
 pass  
  
def consultarNomes(self):  
 print('Consultar por nome')  
 procurarNome = self.lb\_nomeCliente.text()  
  
 try:  
 mydb = mysql.connector.connect(  
 host=host,  
 user=user,  
 password=password,  
 database=database  
  
 )  
 mycursor = mydb.cursor()  
 sql = f"SELECT \* FROM cliente WHERE razaoSocial LIKE '{procurarNome}%' ORDER BY razaoSocial"  
 mycursor.execute(sql)  
 myresult = mycursor.fetchall()  
 print(myresult)  
 df\_cliente = pd.DataFrame(myresult,  
 columns=['idCliente', 'razaoSocial', 'nomeFantasia', 'classificacao', 'cnpjcpf',  
 'ierg', 'endereco', 'numero', 'bairro', 'cep', 'cidade', 'estado',  
 'telefone', 'email', 'observacao', 'prazo', 'usuarioCadastro',  
 'auditoria'])  
 self.all\_data = df\_cliente  
 # *Todo: Carrega o arquivo na Tabela cliente* numRows = len(self.all\_data.index)  
 self.tb\_cliente.setColumnCount(len(self.all\_data.columns))  
 self.tb\_cliente.setRowCount(numRows)  
 # self.tb\_cliente.setHorizontalHeaderLabels(self.all\_data.columns) #Setar o título e cabeçalho da tabela  
  
 for i in range(numRows):  
 for j in range(len(self.all\_data.columns)):  
 self.tb\_cliente.setItem(i, j, QTableWidgetItem(str(self.all\_data.iat[i, j])))  
  
 self.tb\_cliente.resizeColumnsToContents()  
 self.tb\_cliente.resizeRowsToContents()  
  
 mycursor.close()  
  
 except:  
 self.all\_data = 0  
 msg = QMessageBox()  
 msg.setIcon(QMessageBox.warning)  
 msg.setText("Error")  
 msg.setInformativeText("Não foi possível consultar a tabela!")  
 msg.setWindowTitle("Error")  
 msg.exec()  
 pass  
  
  
def avancarTela(self):  
 print('')  
 self.frameTab2.show()  
  
  
def limpaTela(self):  
 self.lb\_razaoSocial.clear()  
 self.lb\_nomeFantasia.clear()  
 self.lb\_class.clear()  
 self.lb\_cnpjcpf.clear()  
 self.lb\_ierg.clear()  
 self.lb\_endereco.clear()  
 self.lb\_numero.clear()  
 self.lb\_bairro.clear()  
 self.lb\_cidade.clear()  
 self.lb\_cep.clear()  
 self.lb\_uf.clear()  
 self.lb\_telefone.clear()  
 self.lb\_email.clear()  
 self.lb\_observacao.clear()  
 self.lb\_prazo.clear()  
  
def incluirNovoCliente(self):  
 self.consultarGeral()  
 self.tabWidget.setCurrentIndex(0)  
 self.limpaTela()