Portfólio – 2º Semestre

Matéria – Análise Orientada a Objetos

Antes de tudo, iniciamos com o Cliente.

Iniciaremos com o primeiro Ator que será o Cliente. Sua construção será da maneira abaixo.

Cliente

A classe Cliente vai ficar assim:

Nome da Classe = Cliente

Atributos:

Cpf = cpf:long

Nome = nome: string

Email = email: string

Telefone = tel: string

Endereço = endereço: string

Os métodos serão:

Consultar se estou cadastrado: consultarCadastro(): bool

Consultar Cliente = consultarCliente(): string

Cadastrar Cliente = cadastrarCliente(): void

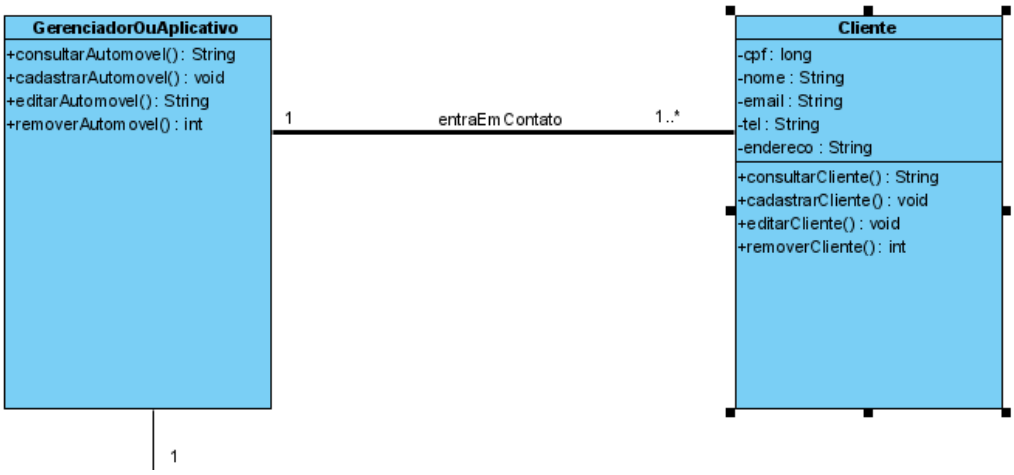
Editar Cliente = editarClienteI(): void

Remover Cliente = removerCliente(): int

A multiplicidade será feita da seguinte maneira:

A classe Cliente poderá ter 1 Cliente ou vários que está para 1 GerenciadorOuAplicativo, poderá ter somente um GerenciadorOuAplicativo por vez.

Veja a imagem abaixo:



A empresa tem muitos automóveis. Nesse caso criamos uma classe para a empresa, ou se não, criamos um gerenciador que poderá ser uma pessoa ou aplicativo para esta empresa, no qual terá acesso ao sistema.

O segundo possível Ator poderá ser o aplicativo, ou bot ou até mesmo uma pessoa física será o Gerenciador ou Aplicativo, sua construção será da seguinte maneira, por poder ser um bot ou aplicativo, não foi colocado atributos específicos para esse ator. Veja abaixo:

Nome da Classe = GerenciadorOuAplicativo

Não possuirá inicialmente atributos, apenas métodos. Os métodos serão.

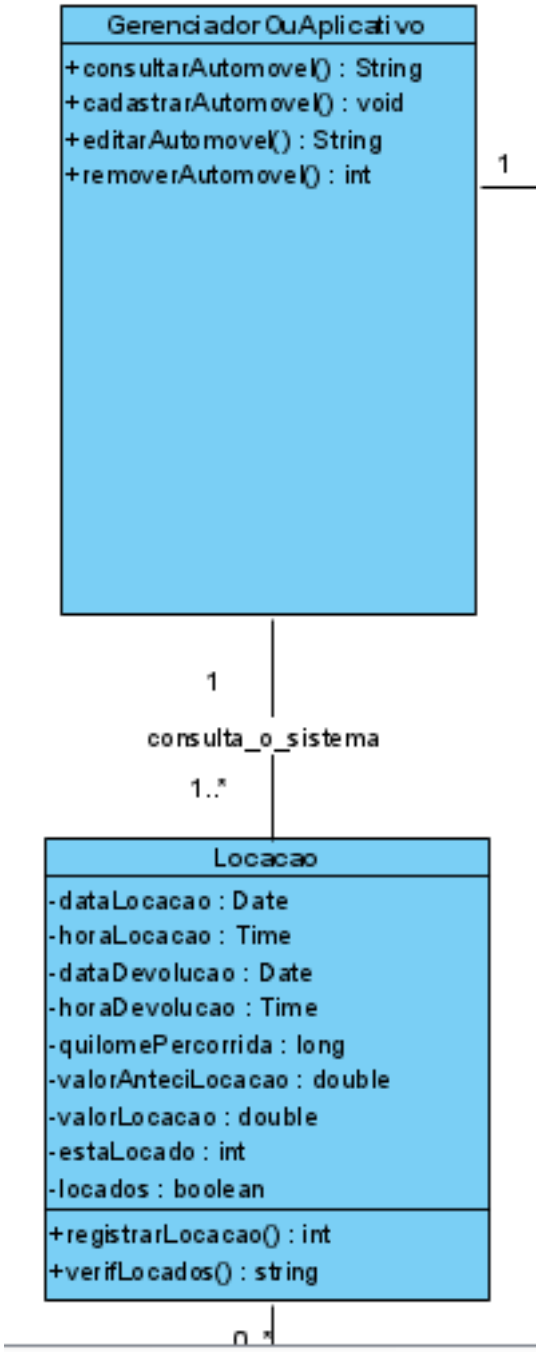
consultarAutomovel(): string

cadastrarAutomovel(): void

editarAutomovel(): string

removerAutomovel(): int

A multiplicidade entre GerenciadorOuAplicativo foi detalhada acima, agora detalharemos a multiplicidade entre GerenciadorOuAplicativo e Locacao. A multiplicidade será de 1 GerenciadorOuAplicativo para 1 ou muitas Locacao, ou seja, um GerenciadorOuAplicativo poderá fazer diversas requisições de Locacao. Veja a imagem abaixo:



Logo em seguida criamos uma classe para acessar o sistema seja pelo web app ou o app mesmo. Essa classe terá o acesso ao sistema, ou seja, será a interface do nosso produto.

Classe Locação será construída da seguinte maneira:

Nome da classe = Locacao

Atributos da classe:

Data da locação = dataLocacao: Date

Hora da locação = horalocacao: Time

Data da Devolução = dataDevolucao: Date

Hora da Devolução = horaDevolucao: Time

Quilometragem percorrida = quilomePercorrida: long

Valor antecipado da Locação = valorAnteciLocacao: double

Valor da locação = valorLocacao: double

Se foi devolvido ou não = estaLocado: int

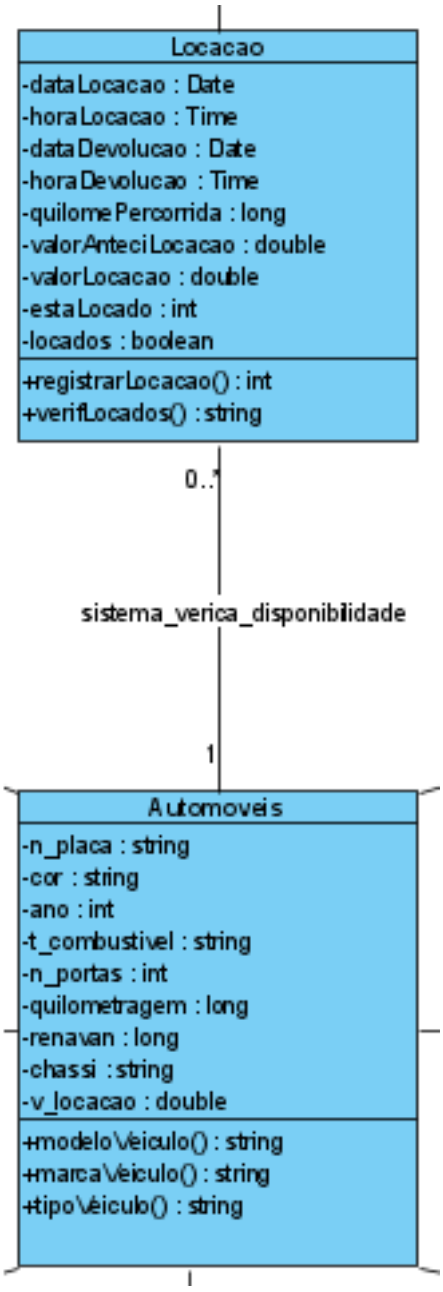
Quais estão locados = locados: boolean

Os métodos serão:

Registrar locação = registrarLocacao(): int

Verificar quais estão locados = verifLocados(): string

A multiplicidade entre Locacao e Automoveis será dada da seguinte forma. Uma Locacao ou varias Locacao fará requisição para 1 Automeveis por vez, ou seja, poderá fazer diversas Locacao, mas para cada requisição poderá somente 1 Automoveis por vez. Veja o exemplo abaixo:



Criar classe com nome Automoveis para representar a classe dos automóveis ao todo no qual servirá para herança de outras classes. Essa classe terá acesso a todas as outras, ou seja, será nossa classe pai ou o sistema todo em si.

Classe Automóveis será da seguinte forma:

Nome da classe = Automoveis

Atributos da classe:

Número da placa = n_placa: string

Cor = cor: string

Ano = ano: int

Tipo de combustível = t_combustivel: string

Número de portas = n_portas: int

Quilometragem = quilometragem: long

Renavan = renavan: long

Chassi = chassi: string

Valor da locação = v_locacao: double

Os métodos serão:

Modelo do veículo = modeloVeiculo(): string

Marca do veículo = marcaVeiculo(): string

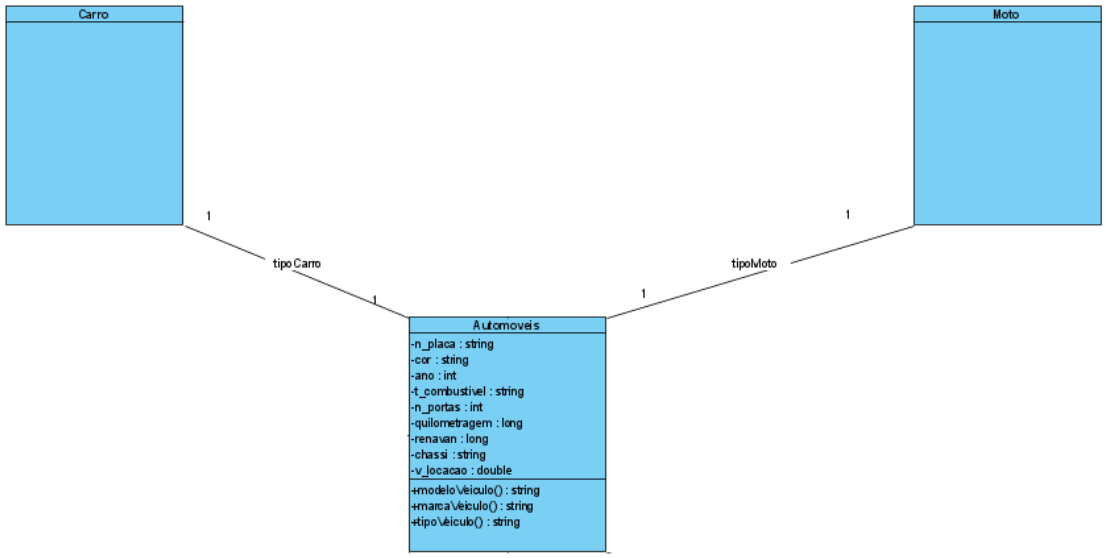
Tipo do veículo = tipoVeiculo(): string

Os automóveis herdaram os atributos da classe Automoveis, número da placa, cor, ano, tipo de combustível, número de portas, quilometragem, RENAVAM, chassi, valor de locação.

A classe Automveis, será o sistema ao todo, ou seja, terá acesso a todas as outras classes. Sua multiplicidade será feita da seguinte forma:

Automoveis é a classe pai, dela criaremos todo o sistema para que as outras herdaram.

1- Automoveis classe pai, dela teremos a classe filha Carro, a classe filha Moto. Sua multiplicidade será dada de 1 para um, ou seja, a classe Automoveis poderá fazer requisição somente de 1 de cada vez, Automveis 1 – Carro 1, Automoveis 1 – Moto 1, não poderá fazer fazer varias requisições ao mesmo tempo. Veja a imagem abaixo:



Marca vai ser assim:

Nome da classe = Marca

Atributo:

Descrição da Marca = descMarca: string

Receber quais modelos reberam essa marca = quaisModRebMar: string

Método:

Consultar Marca = consultarMarca(): string

Quais modelos possuem essa marca = modPossuiMarca(): string

Modelo vai ser assim:

Nome da classe = Modelo

Receber uma marca especifica = marcaEspecifica: string

Atributo:

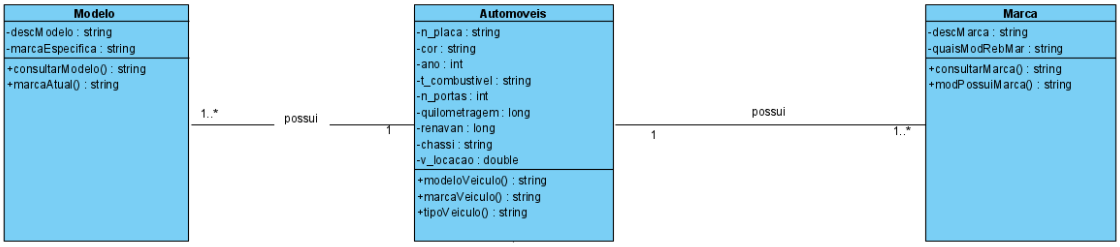
Descrição do Modelo = descModelo: string

Qual marca foi usada no modelo = marcaAtual(): string

Método:

Consultar Modelo = consultarModelo(): string

A classe pai Automoveis terá sua multiplicidade da seguinte forma com as duas classes Modelo e marca, será dada de varios Modelos para cada Automoveis e várias Marcas. Veja a imagem abaixo:



Cada carro tem:

Modelo = modeloCarro

Marca = marcaCarro

Criando os fatores de multiplicidade.

Cada carro tem um modelo e uma marca:

Será dado para cada carro o método consultarModelo e consultarMarca, pois trabalharemos com herança.

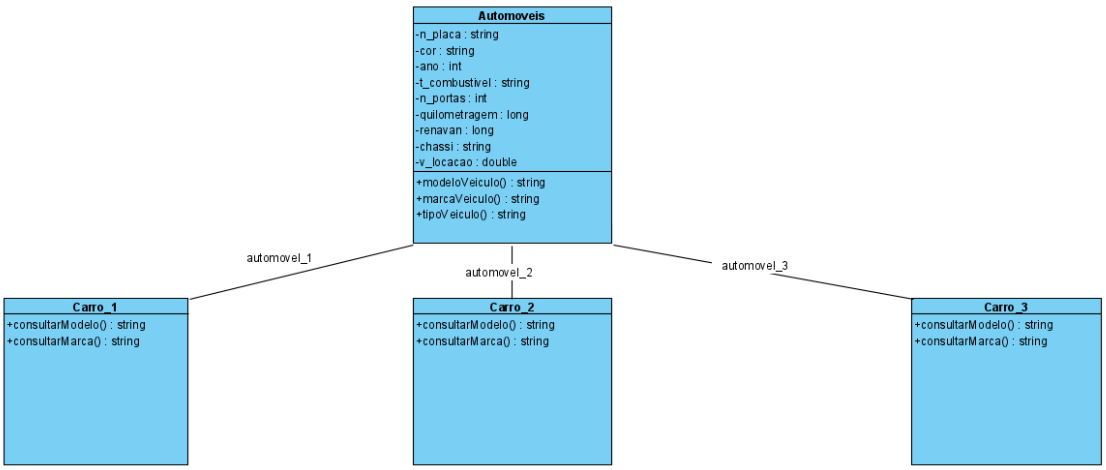
Cada carro tem um modelo e uma marca.

Um modelo pode se relacionar-se a muitos carros.

Uma marca pode referir-se a muitos modelos.

Porém cada modelo só tem uma marca.

E nosso ultimo exemplo será a multiplicidade entre Automoveis com suas classes filhas que será geradas após o cadastro de cada Automoveis disponíveis para Locacao.



Um carro pode ser alugado por muitos clientes.