UNOCHAPECÓ

VINÍCIUS POZZAN, LUIS HENRIQUE SILVEIRA BRAATZ, IGOR MATHEUS CARARO DE JESUS, RICARDO SCHINEMEIER, LUIZ EDUARDO BARELLA

TRABALHO FINAL ENGENHARIA DE SOFTWARE II

Chapecó

2022

DESCRIÇÃO TEXTUAL DOS CASOS DE USO REFERENTE A QUESTÃO PROPOSTA "FÁBRICA DE CONCRETO”

- O cliente realiza o pedido ao representante, informando o tipo e quantidade desejadas.
- O representante registra o pedido e encaminha para o encarregado da produção.
- O encarregado da produção verifica a disponibilidade da matéria prima.
- O encarregado de compras verifica a disponibilidade da matéria prima no estoque e encaminha para a indústria. Se a quantidade em estoque não suprir o necessário para a produção do pedido , realiza a compra de mais matéria prima.
- A indústria inicia a produção do concreto e quando finalizado, envia para a transportadora.
- A transportadora realiza a entrega do pedido ao cliente.
- O cliente realiza o pagamento ao setor financeiro e recebe o pedido.
- O setor financeiro recebe o pagamento.

DIAGRAMA CASOS DE USO UML "FÁBRICA DE CONCRETO”

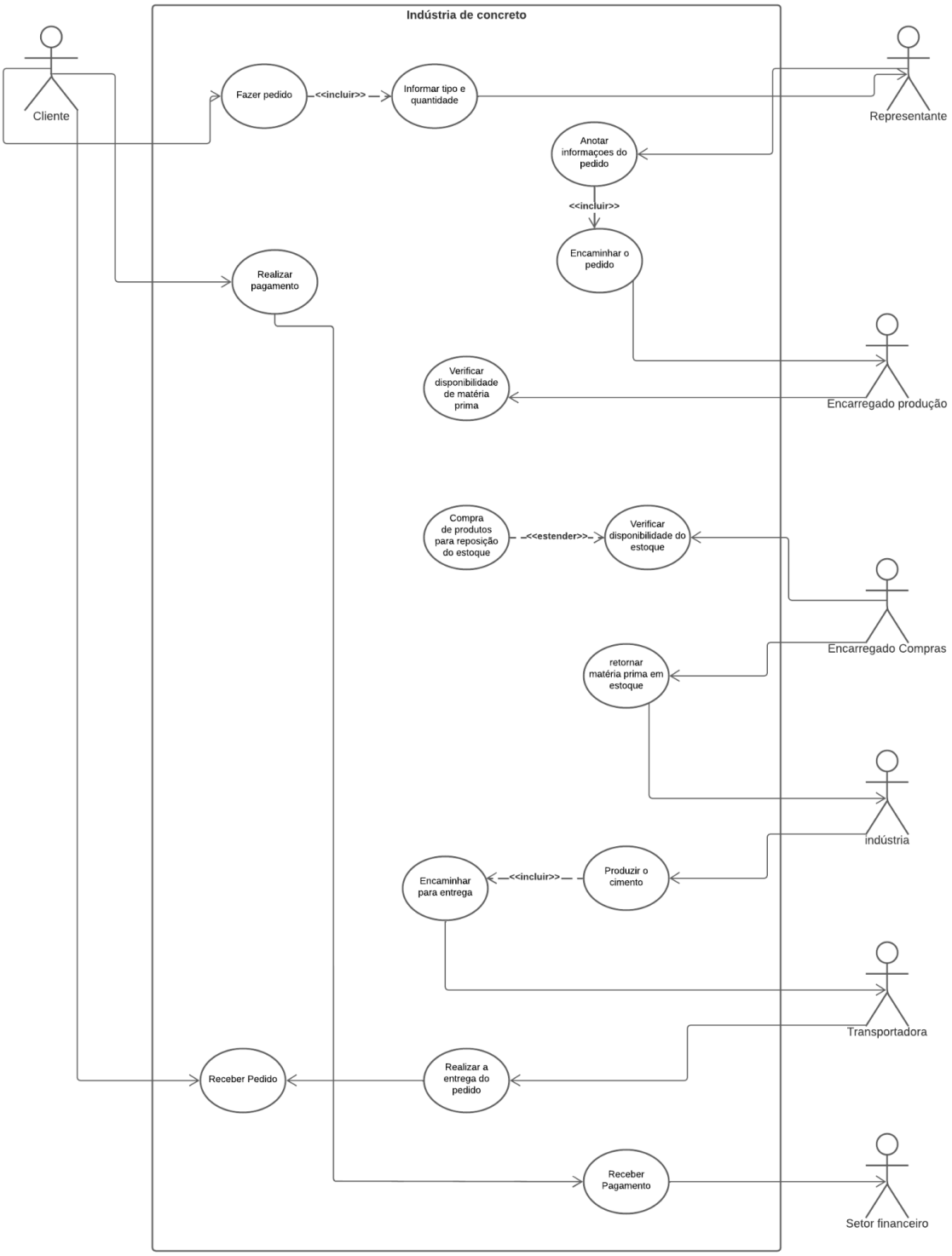


DIAGRAMA DE DOMÍNIO UML "FÁBRICA DE CONCRETO”

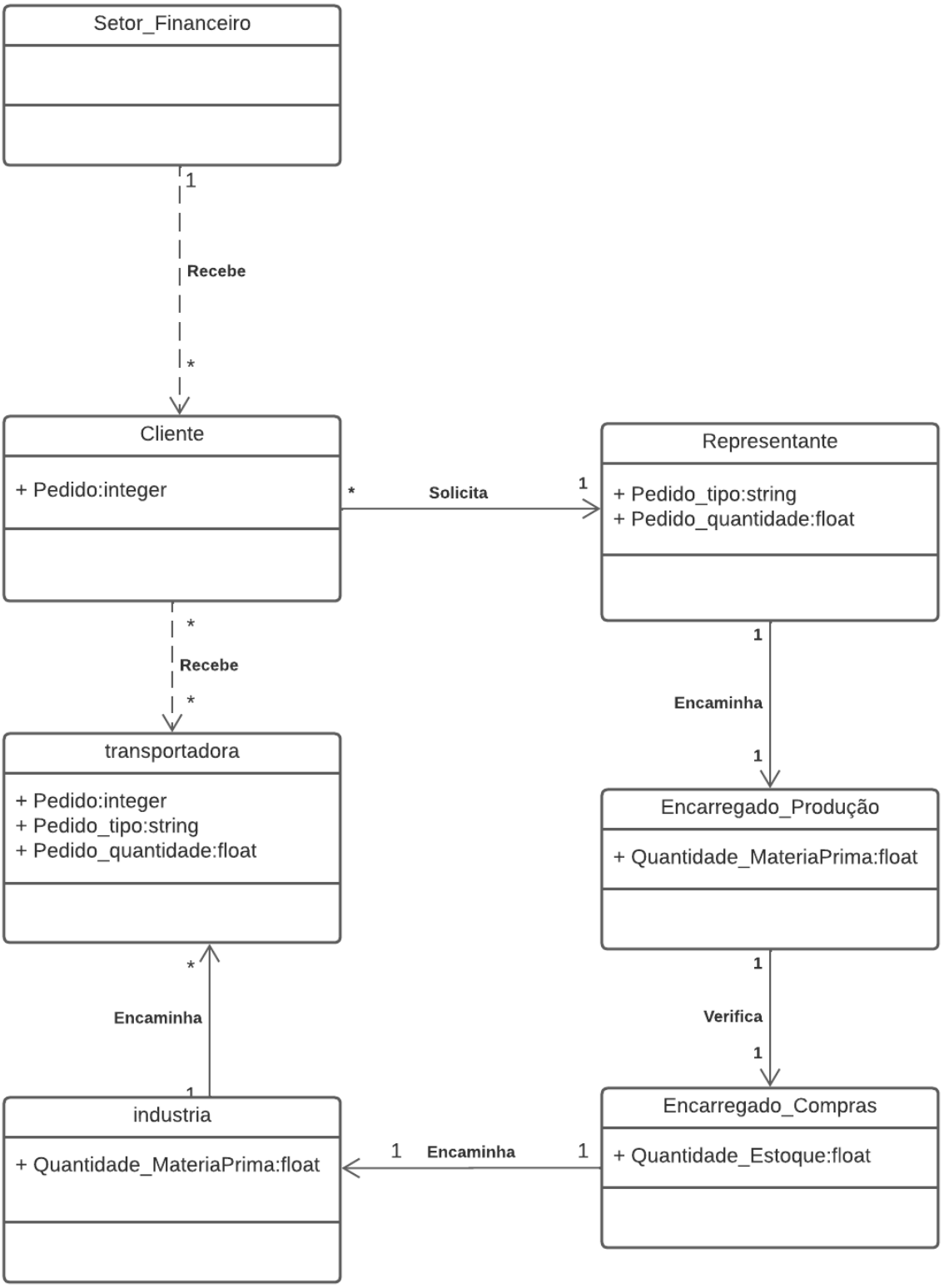
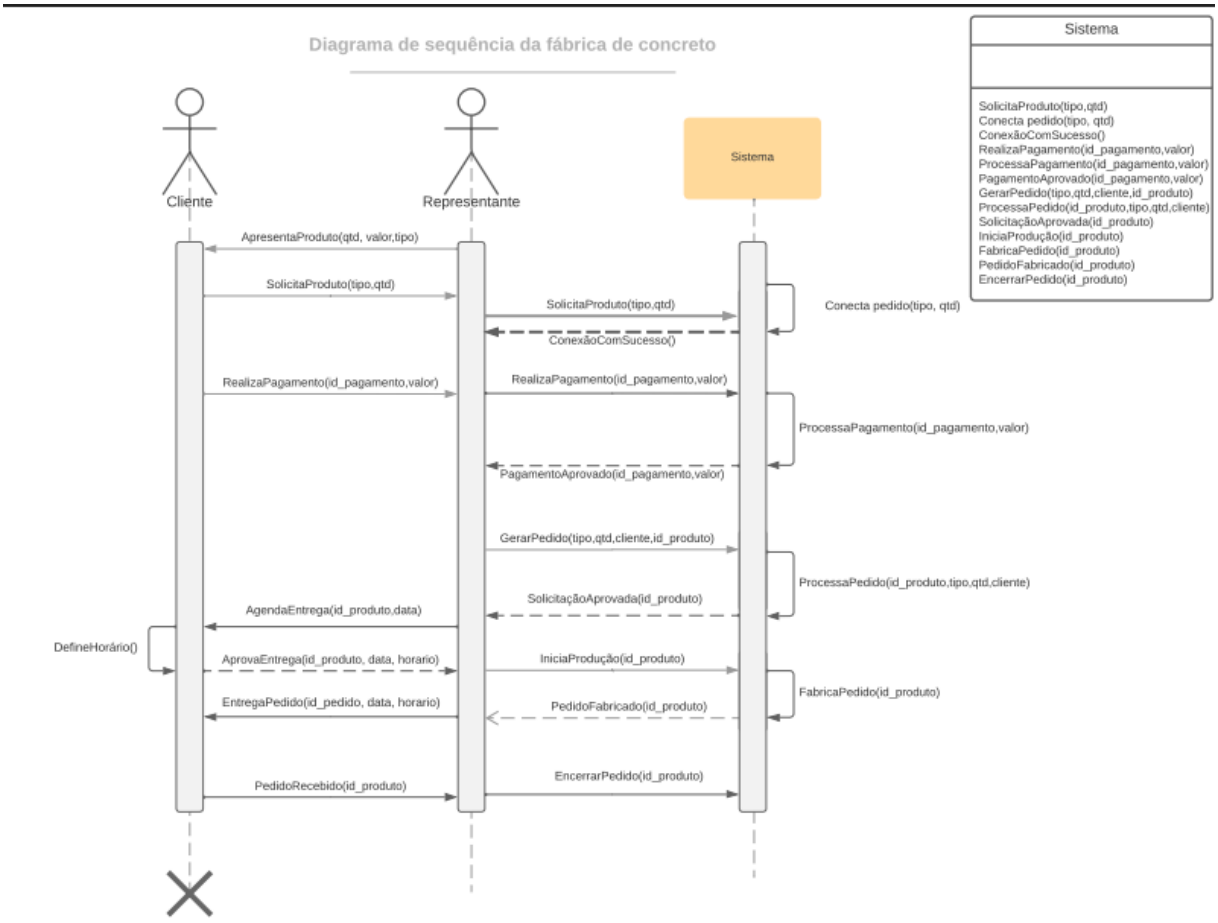


DIAGRAMA DE SEQUÊNCIA CASOS DE USO



CONTRATOS DAS OPERAÇÕES DO DSS

Operação: criarPedido(tipo,qtd)

Caso de Uso: Pedido cliente

Controlador: Sistema

Pré-condição: nenhuma

Pós-condição: - uma instância de Pedido P foi criada.

- P se associou com Sistema.

- Atributos de P foram inicializados.

Operação: gerarPedido(id_produto: id_produto, tipo: qtd)

Caso de Uso: Pedido cliente

Controlador: Sistema

Pré-condição: está se processando um pedido

Pós-condição: - uma instância de FabricaPedido Fp foi criada.

- Fp se associou com a Pedido atual.

- Fp se associou com o Pedido cujo código é id_produto.

Operação: PedidoFabricado()

Caso de Uso: Pedido cliente

Controlador: Sistema

pré-condição: pedido foi fabricado.

Pós-condição: - Encerrar produção.

- Entrega pedido.

CLASSES DIAGRAMA DE IMPLEMENTAÇÃO

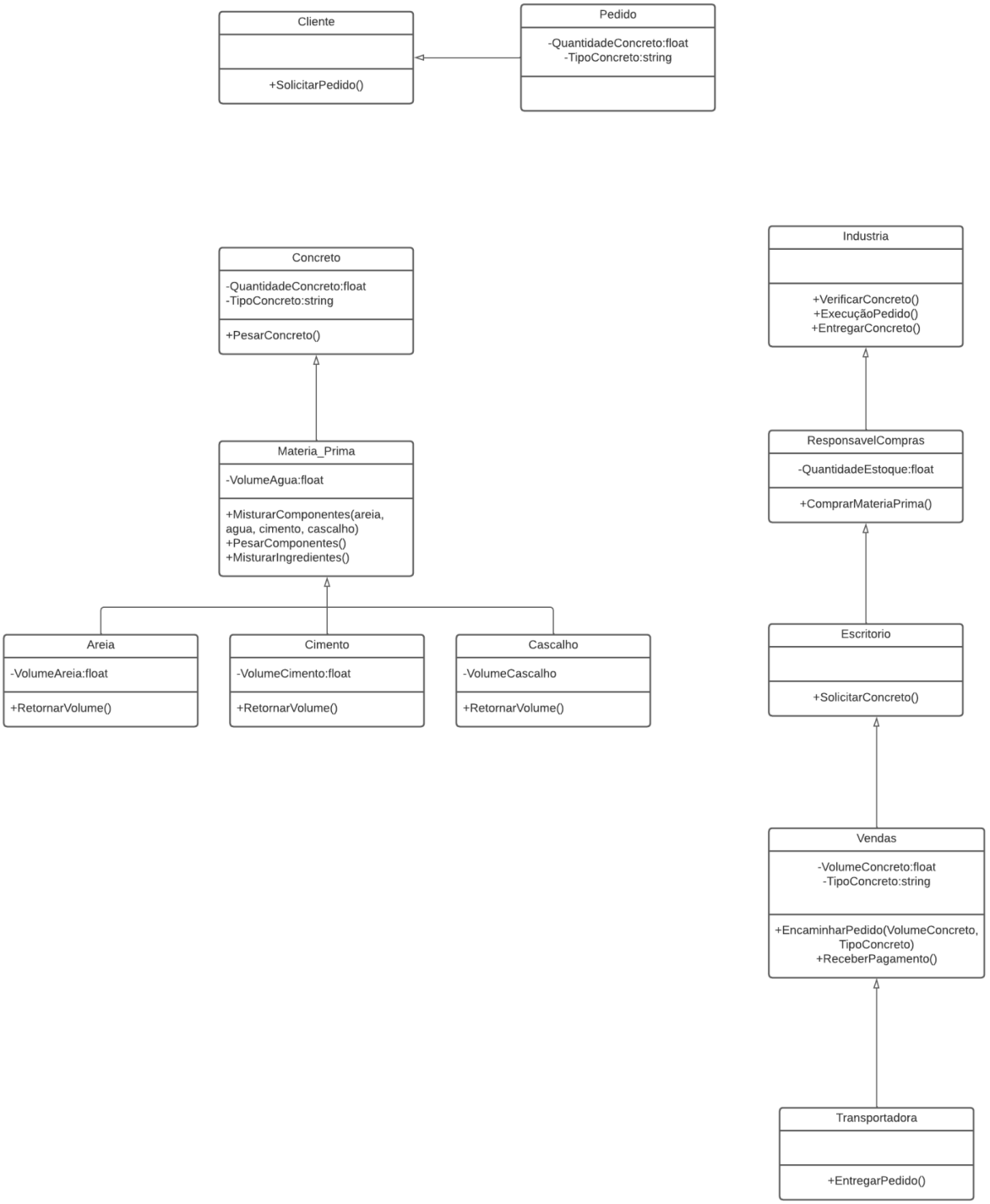


DIAGRAMA DE CLASSES IMPLEMENTADO EM PYTHON

```
 class Cliente:
    def __init__(self, nome, fone, endereco):
        self.nome = name
        self.fone = fone
        self.endereco = endereco

    def RealizarPedido(self, Pedido):
        return "{} pedido {}".format(self.nome, Pedido)

class Pedido:
    def __init__(self, TipoConcreto, QuantidadeConcreto):
        self.TipoConcreto = TipoConcreto
        self.QuantidadeConcreto = QuantidadeConcreto

class Concreto:
    def __init__(self, TipoConcreto, QuantidadeConcreto):
        self.TipoConcreto = TipoConcreto
        self.QuantidadeConcreto = QuantidadeConcreto

class MateriaPrima:
    def __init__(self, VolumeAgua):
        self.VolumeAgua = VolumeAgua

    def MisturarComponentes(self, Misturar):
        return "{} Mistura {}".format(self.VolumeAgua, Misturar)



class Cascalho(MateriaPrima):
    def __init__(self, VolumeCascalho):
        super().__init__(Materia_prima, VolumeAgua)
        self.VolumeCascalho = VolumeCascalho

    def Volume(self, Volume):
        return "{} Volume {}".format(self.VolumeCascalho,Volume)
```

```
class Cimento(MateriaPrima):
    def __init__(self, VolumeCimento):
        super().__init__(Materia_prima, VolumeAgua)
        self.VolumeCimento = VolumeCimento

    def Volume(self, Volume):
        return "{} Volume {}".format(self.VolumeCimento,Volume)



class Areia(MateriaPrima):
    def __init__(self, VolumeAreia):
        super().__init__(Materia_prima, VolumeAgua)
        self.VolumeAreia = VolumeAreia

def Volume(self, Volume):
        return "{} Volume {}".format(self.VolumeAreia,Volume)



class ResponsavelCompras:
    def __init__(self, QuantidadeEstoque):
        self.QuantidadeEstoque = QuantidadeEstoque

    def ComprarMateriaPrima(self, Comprar):
        return "{} Compra {}".format(self.QuantidadeEstoque, Compra)

class Vendas:
    def __init__(self, VolumeConcreto, TipoConcreto):
        self.VolumeConcreto = VolumeConcreto
        self.TipoConcreto = TipoConcreto

    def EncaminharPedido(self, Encaminhar):
        return "{} Encaminhar {}".format(self.TipoConcreto,Encaminhar)

class Escritorio:
    def __init__(self):
```

```
        def SolicitarConcreto(self, Solicitar):
         return "{} Solicitar{}".format(self.QuantidadeConcreto,Solicitar)

class Transportadora:
    def __init__(self):

        def EntregarPedido (self, Entregar):
            return "{} Entregar {}".fromat(self.TipoConcreto)

class industria:
    def __init__(self):

        def VerificarConcreto (self, VerificarConcreto):
            return "{} Verificar {}".fromat(self.TipoConcreto)

        def ExecutarPedido(self, Executar):
            return "{} Executar {}".fromat(self.TipoConcreto)
```

MODELO DE INTERFACE/FORMULÁRIO

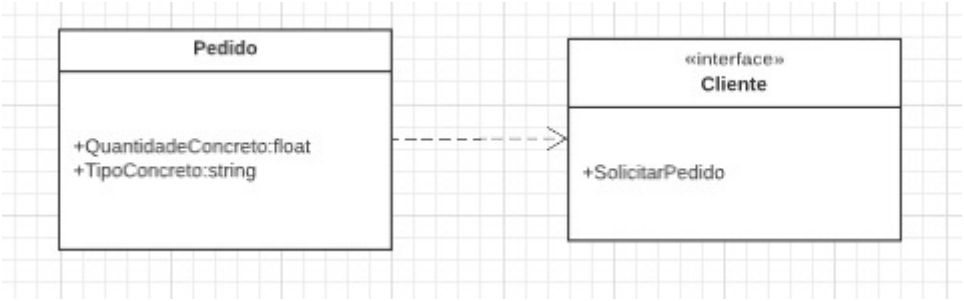


DIAGRAMA DE COMUNICAÇÃO COM PADRÕES “GRASP”