

## **DCC-UFMG Ensino à Distância**

### **Estudo de Caso**

#### **Descrição 1**

Virtual LTDA é o assunto do nosso estudo de caso que se estenderá durante o curso. Uma empresa pequena localizada em Belo Horizonte, Minas Gerais, a Virtual LTDA é especializada em equipamentos musicais (guitarra, violão, etc). Virtual LTDA está adicionando a sua linha de produtos, outros produtos. Incluindo tecnologias de mixagem e gravação, microfones e CD *players*. A Virtual LTDA é uma empresa ambiciosa, ela deseja abrir uma filial em São Paulo dentro de um ano.

Algumas de suas vendas originam-se de organizações comerciais. Os fundadores da Virtual LTDA, acreditam que não podem efetivamente administrar sua empresa com um sistema de cobrança e cadastro de pedidos inadequados. Os sócios não são especialistas em tecnologia de informação, mas eles sabem que o sucesso depende da extensibilidade do sistema que será desenvolvido. A preocupação inicial é a aplicabilidade da Internet à seus negócios. Eles querem minimizar os custos com a tecnologia usada enquanto mantêm a flexibilidade na troca de plataforma. Eles fazem as seguintes observações sobre a empresa:

- Muitos de seus produtos são muito caros. Neste caso os balconistas têm participação nas vendas.
- Muitos dos clientes querem ser capazes de obter informação de um determinado produto sem interagir com um representante da empresa.
- Muitos clientes querem pedir outros produtos a partir da Virtual LTDA, os quais não são oferecidos, incluindo microfone, tecnologia de mixagem, etc.

Eles acreditam que uma solução baseada na Internet poderá beneficiar a Virtual LTDA. Nosso desafio é projetar um sistema que atenda as necessidades imediatas da empresa e que seja flexível o suficiente para suportar outros tipos de produtos no futuro.

#### **Descrição 2**

No início de cada semestre os alunos devem requisitar um catálogo de cursos contendo os cursos oferecidos no semestre. Este catálogo deve conter informações a respeito de cada curso tais como: professor, departamento e pré-requisitos. Desse modo, os alunos podem tomar suas decisões mais apropriadamente.

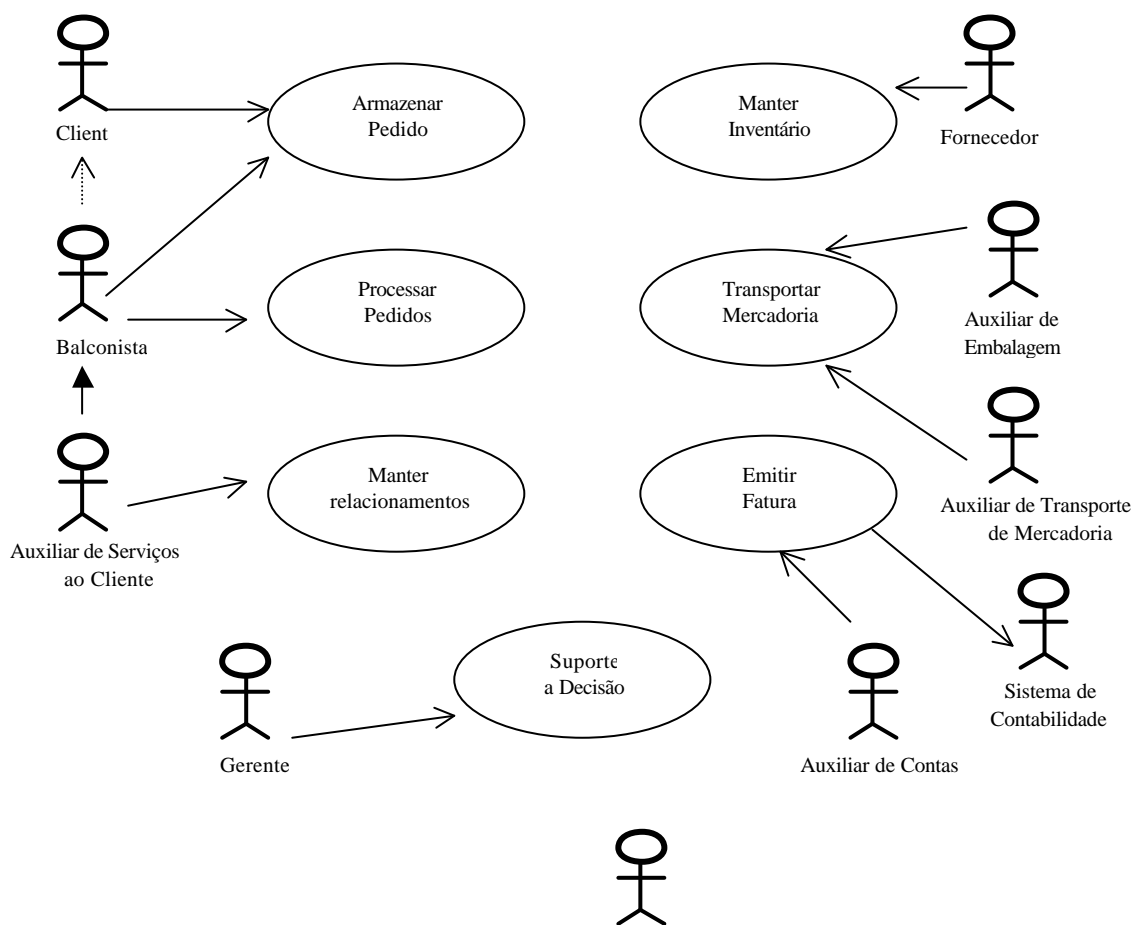
O novo sistema permitirá que os alunos selecionem quatro cursos oferecidos para o próximo semestre. Além disso, o aluno indicará dois cursos alternativos, caso o aluno não possa ser matriculado na primeira opção. Cada curso deverá ter no máximo de 10 e mínimo de 3 alunos. Um curso com o número de alunos inferior a 3 será cancelado. Para cada matrícula feita por um aluno, o sistema envia informação ao sistema de cobrança para que cada aluno possa ser cobrado durante o semestre.

Os professores devem acessar o sistema “on-line”, indicando quais cursos irão lecionar. Eles também podem acessar o sistema para saber quais alunos estão matriculados em cada curso. Em cada semestre, há um prazo para alteração de matrícula, que corresponde

ao período de matrícula. Os alunos devem poder acessar o sistema durante esse período para adicionar ou cancelar cursos.

## 1. Diagrama de Caso de Uso

É um diagrama usado para se identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descreve o sistema, seu ambiente e a relação entre os dois. Os casos de uso são usados para obter requisitos a partir da perspectiva do usuário. Os componentes deste diagrama são os Atores e os Casos de Uso. Abaixo é mostrado um diagrama de Caso de Uso da Virtual LTDA.



### a. Evento

Eventos podem ser definidos como ações ou estímulos que causam mudanças no sistema e exigem alguma reação do sistema. Os eventos podem ser classificados em duas categorias: eventos externos e eventos internos.

- i. **Eventos externos:** São os mais fáceis de encontrar. Eles são qualquer estímulo ao sistema que origina a partir de fora dos limites do sistema. Por exemplo, a solicitação de um pedido pelo

cliente. Para identificar eventos externos, focalize nos eventos e faça perguntas, quem e o que esta estimulando o evento?

- ii. **Eventos internos:** São mais diversos. Um tipo de evento interno é um temporizador que causa o sistema fazer alguma coisa. Novamente focalize nos eventos e faça perguntas, quem e o que esta estimulando o evento?

#### b. Ator



Representa um Ator.

Representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, outro sistema, etc. Algumas de suas características são descritas abaixo:

- Um ator é representado por um boneco, o qual é rotulado com o nome do papel. O ator descreve o papel que um usuário assume com relação ao sistema.



Cliente

- Ator não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar.
- Ator pode interagir ativamente com o sistema.
- Ator pode ser um receptor passivo de informação
- Ator pode representar um ser humano, uma máquina ou outro sistema.

#### c. Papel

Podemos definir um papel como sendo o personagem representado por um ator. Por exemplo, em uma peça de teatro você como ator poderia representar ou o papel de bandido ou o papel de policial. Portanto, um ator representa um papel, não um usuário individual do sistema: uma pessoa pode assumir diferentes atores no sistema.



Balconista



Gerente

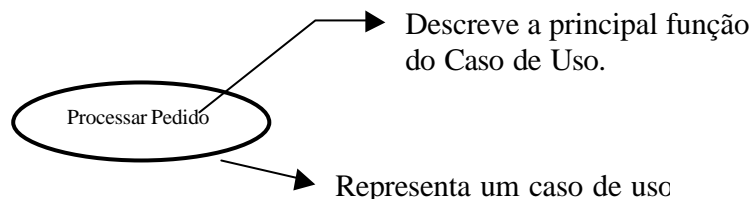


Fornecedor



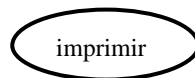
Cliente

#### d. Caso de Uso



Um Caso de Uso (em inglês *use case*) é uma interação típica entre um usuário e um sistema. Um caso de uso captura alguma função visível ao usuário e, em especial, busca atingir uma meta do usuário. Assim um caso de uso pode ser definido como uma sequência de ações que o sistema executa e produz um resultado de valor para o ator. Algumas de suas características são descritas abaixo:

- Um caso de uso é representado por uma elipse, a qual é rotulada com um verbo que representa uma função do sistema.



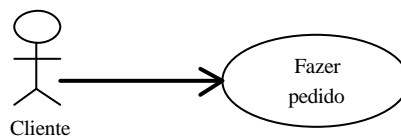
- Um caso de uso modela o diálogo entre atores e o sistema.
- Um caso de uso é iniciado por um ator para invocar uma certa funcionalidade do sistema.
- Um caso de uso é fluxo de eventos completo e consistente.
- O conjunto de todos os casos de uso representa todas as situações possíveis de utilização do sistema.

#### e. Relacionamentos

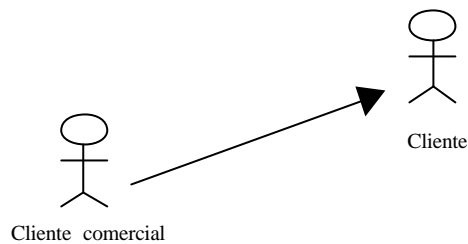
Um relacionamento é definido como uma conexão entre objetos e serve de canal de comunicação entre eles. Geralmente, é representado por uma linha que conecta dois ou mais objetos, podendo ser direcional.

Representam um tipo de conexão entre atores e caso de uso ou entre casos de uso.

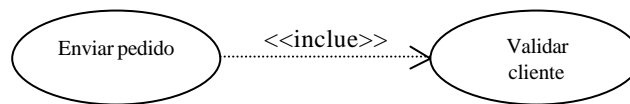
- **associação:** Representamos um relacionamento entre um ator e um caso de uso por uma seta simples. Único relacionamento entre atores e casos de uso.



- **Generalização:** Representamos um relacionamento de herança entre atores como mostrado abaixo.



- **<<include>>**: um relacionamento de inclusão entre dois casos de uso significa que um caso de uso base incorpora explicitamente o comportamento de outro caso de uso. Uma relação de inclusão se representa usando uma seta pontilhada e rotulada com a palavra *include*.



O relacionamento acima poder ser lido do seguinte modo: o caso de uso Enviar pedido include o caso de uso Validar Cliente.

Quando deve ser usado o relacionamento de extensão:

- Comportamentos comuns em diferentes casos de uso.
  - Necessidade de melhorar o entendimento de um caso de uso: gerencia redundância e flexibiliza mudanças.
  - O relacionamento de inclusão permite a um caso de uso, incluir o fluxo de eventos especificado em um outro caso de uso.
- **<<estende>>**: um relacionamento de extensão indica que um caso de uso base incorpora implicitamente o comportamento de outro caso de uso. Um caso de uso pode estender somente em certos pontos, chamados pontos de extensão. Uma relação de extensão se representa usando uma seta pontilhada e rotulada com a palavra *estende*. Um relacionamento de extensão é usado para modelar a parte de um caso de uso que o usuário pode ver como o comportamento opcional do sistema. Desta forma se separa o comportamento opcional do comportamento obrigatório.



O relacionamento acima poder ser lido do seguinte modo: o caso de uso Enviar pedido parcial estende o caso de uso Enviar pedido.

Como funciona o relacionamento de extensão :

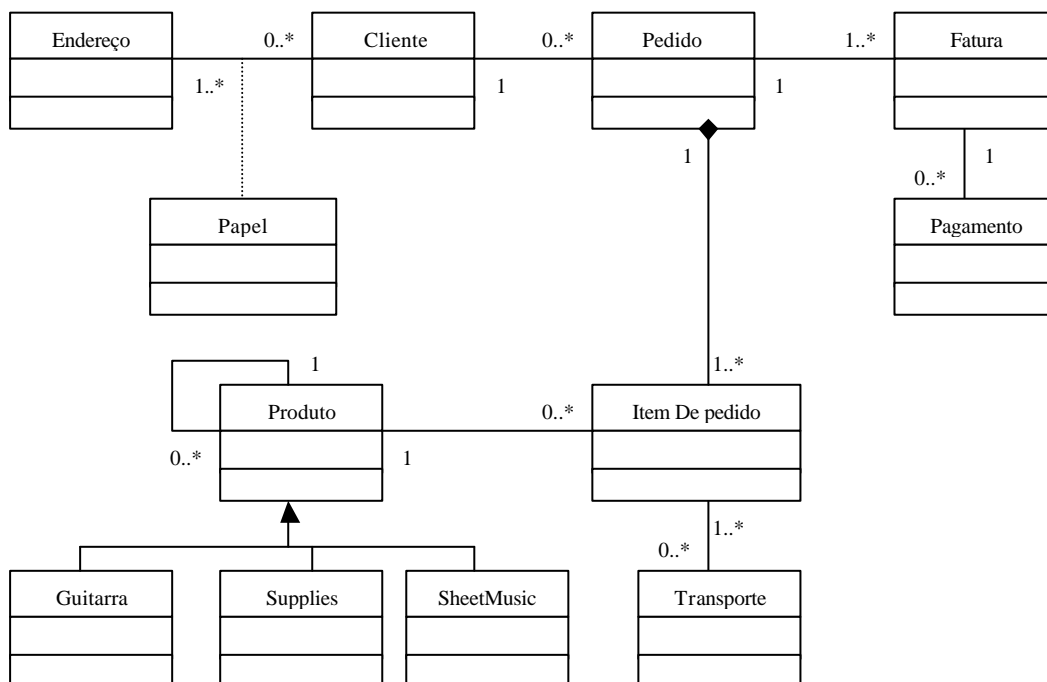
- Em um ponto de extensão, sobre certas circunstâncias, o comportamento estendido é executado.
- O controle é retomado para o caso de uso base no mesmo ponto onde a extensão foi executada.
- Cada ponto de extensão deve ter um nome único no caso de uso base.

Quando deve ser usado o relacionamento de extensão:

- Para adicionar novos comportamentos sob certas condições, ou seja, um comportamento opcional do sistema.
- Incorporar requisitos funcionais específicos que não fazem parte do fluxo do caso de uso base.

## 2. Diagrama de Classes

O diagrama de classes descreve os vários tipos de objetos que existem no sistema e os relacionamentos estáticos que existem entre eles. O diagrama de classes também mostra os atributos (propriedades) e as operações (métodos) de uma classe. Diagramas de classe permitem, planejar como as classes/objetos funcionará e interagirão. A figura abaixo mostra o diagrama de classes para a Virtual LTDA.



Uma classe é representada como mostrada abaixo:

Nome da classe
Atributos da classe
Operações da classe

Uma classe é uma descrição de um conjunto de objetos que partilham os mesmos atributos, operações, relações e semântica. Por exemplo, O cliente “João da Silva” pode ser considerado um objeto relevante num sistema que pretende manipular informação referente aos clientes de uma empresa.

*Atributo da classe:* são propriedades que permite identificar uma classe ou um objeto. O João da Silva além do nome, também é caracterizado por outros atributos, nome, endereço, número do contribuinte, CPF, etc.

*Operações da Classe:* O João da Silva possui uma identidade própria, isto é, para a empresa, ele é distinto de todos os outros clientes. Sobre o cliente João da Silva podem-se efetuar várias operações, nomeadamente emitir-lhe faturas, efetuar alterações de endereço, elimina-lo do sistema.

O João da Silva relaciona-se com a empresa através, por exemplo, dos produtos ou serviços que adquire. Provavelmente existirão outros clientes na empresa, e todos eles são caracterizados pelo mesmo conjunto de atributos, pelas mesmas operações e relações e todos são distintos um dos outros. Esses diversos clientes podem ser agrupados em uma classe, a classe dos clientes da empresa. Note-se que os objetos não têm necessariamente que corresponder a entidades humanas ou, mais genericamente, a entidades com representação física (por exemplo, uma fatura).

#### a. Tipos de classes

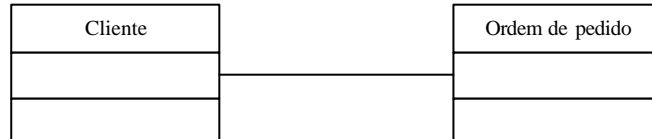
- i. Entidade: É uma classe que modela objetos cuja informação e o comportamento associado são, de maneira geral, persistentes. No presente estudo de caso, as classes de objetos: clientes, produtos, são exemplos de classes de entidade.
- ii. Fronteira: classes de fronteira servem como fronteira entre os atores externos desejando interagir com a aplicação e a classe de entidade. Muitas classes de fronteira são componentes da interface do usuário, as quais tomam a forma de um formulário ou tela usados para interagir com a aplicação. Portanto classe de fronteira é uma classe que modela a comunicação entre a vizinhança do sistema e suas operações. Exemplos: interface do tipo janela, protocolo de comunicação, interface de impressão, etc.
- iii. Controle: classes de controle são coordenadoras da atividade no domínio da aplicação. Tipicamente, uma classe de controle pode

assumir um dos vários papéis: como comportamento relacionado a transações, um serviço que separa os objetos de entidade a partir dos objetos de fronteira. Basicamente, uma classe de controle, é uma classe que modela o comportamento de controle específico para uma ou mais Caso de Uso. Suas principais características são:

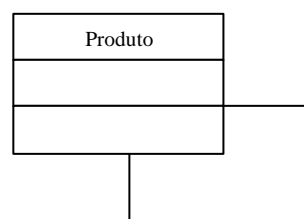
- Cria, ativa e anula objetos controlados.
- Controla a operação de objetos controlados.
- Controla a concorrência de pedidos de objetos controlados.

## b. Relacionamentos

- Associação entre Classes:** o tipo mais comum de relacionamento em UML, uma associação define como os objetos de uma classe são conectados a objetos de outra classe. Sem essa associação nenhuma mensagem pode passar entre objetos da classe em tempo de execução. Existe uma associação entre duas classes se um instância de uma classe deve conhecer sobre a existência da outra de modo a realizar seu trabalho. No diagrama , uma associação é uma linha conectando duas classes.

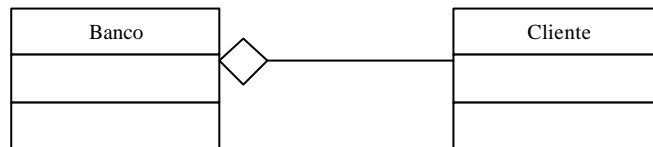


- Associação Reflexiva entre Classes:** algumas vezes uma associação é necessária entre dois objetos da mesma classe. Chamamos esse tipo de associação de associação reflexiva. Isso poderá ser útil, por exemplo, quando um balconista esta vendendo uma guitarra e quer recomendar algum produto relacionado. Para prover essa característica e modelar e modelar –la de acordo com a UML, nós precisamos usar uma associação reflexiva na classe Produto.



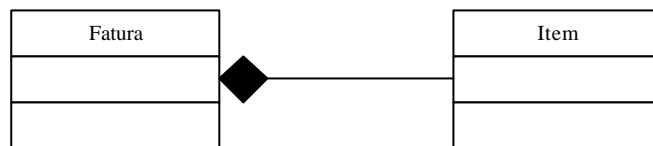


- iii. **Associação de Agregação entre Classes:** As agregações são um caso especial das associações. Uma agregação modela um relacionamento *tem um* (ou *parte de*, no jargão da UML) entre pares. Este relacionamento significa que um objeto contém outro. No contexto de uma agregação, os objetos podem existir independentemente uns dos outros. Nenhum objeto é mais importante do que o outro no relacionamento. Um losango aberto simboliza a agregação. O losango toca o objeto que é considerado o todo do relacionamento: a classe que se refere à outra classe. O todo é constituído de partes.



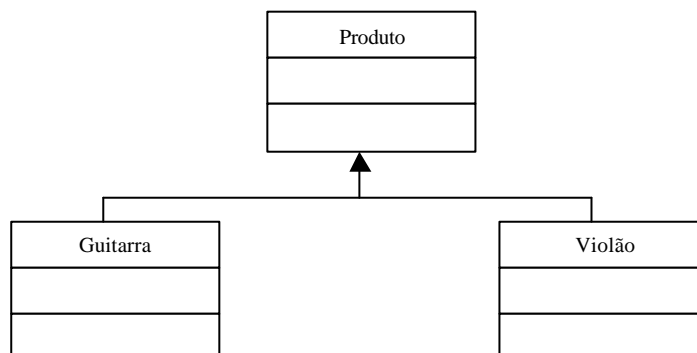
Como o Banco e Cliente são independentes, eles são pares. Você pode dizer que o objeto Banco e o objeto Cliente são pares, porque os objetos Cliente podem existir independentemente do objeto Banco. Isso significa, que se o objeto encerrar suas operações, os clientes não desaparecerão com o banco. Em vez disso, os clientes podem se tornar clientes de outro banco. Do mesmo modo o cliente pode sacar seus fundos e o banco continuará.

- iv. **Associação de Composição entre Classes:** As composições são um caso especial das associações de agregação. Elas representam a noção de composição e apenas têm sentido em associações “um para muitos” e, mais raramente, em associações “muitos para muitos”. Enquanto que nas associações anteriormente referidas não existem classes mais importantes, nas agregações existem uma classe (supra classe) que representa a agregação dos objetos da outra classe (sub classe). A título de exemplo, considere-se os itens (linhas) de uma fatura. Usualmente uma fatura é (ou pode ser) composta por vários itens, cada um deles diz respeito a um produto faturado. Os itens têm atributos próprios (quantidade, preço unitário, descrição do produto, etc.) e entidade própria (é possível distinguir um item de outro na mesma fatura). Uma possível representação deste domínio é o diagrama abaixo. Uma fatura possui vários itens, mas um item apenas diz respeito a uma fatura. Uma agregação tem um losango fechado apontando para a parte contendo o todo.

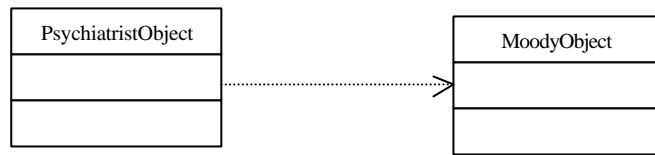


É importante observar que os itens apenas existem enquanto existir a fatura da qual fazem parte. Dito de outra forma, uma fatura é uma agregação, ou seja uma composição de itens. Se removermos uma fatura, os itens dessa fatura também serão removidos. Neste caso temos um tipo de agregação conhecido como **Composição**. Uma composição representa uma relação mais forte entre o objeto agregador e os objetos componentes.

- v. **Generalização de Classes:** uma generalização define uma herança, tal que uma classe refina, isto é, especializa detalhes sobre classe mais gerais. A classe generalizada é frequentemente chamada de *superclasse*, e a classe especializada *subclasse*. Todos os atributos e operações da classe generalizada que tem visibilidade pública e protegida, estão disponíveis para a *subclasse*. Uma generalização tem um triângulo apontando para a superclasse.



- vi. **Dependência entre Classes:** a dependência é um relacionamento no qual a mudança de uma classe pode afetar o comportamento ou estado de outra classe. Tipicamente, dependências são usadas no contexto de classes para mostrar que uma classe usa outra como um argumento na assinatura de uma operação. Ou seja, a dependência indica que um objeto depende da especificação de outro objeto.



Podemos dizer que PsychiatristObject depende de MoodyObject, por dois motivos. Primeiro, o método examine() de PsychiatristObject recebe um MoodyObject como argumento. Segundo, o método examine() chama o método queryMood() de MoodObject. Se o nome ou lista de argumentos do método queryMood() mudar, você precisará atualizar o modo como PsychiatristObject chama o método. Do mesmo modo, se o nome da classe MoodyObject mudar, você terá de atualizar a lista de argumentos do método examine().