Tema 3

Módulo 1

Requisitos de usuário são declarações, em uma linguagem natural com diagramas, de quais serviços o sistema deverá fornecer a seus usuários e as restrições com as quais este deve operar.

Requisitos de sistema são descrições mais detalhadas das funções, serviços e restrições operacionais do sistema de software.

Requisitos funcionais são declarações de serviços que o sistema deve fornecer, de como o sistema deve reagir a entradas específicas e de como o sistema deve se comportar em determinadas situações.

Requisitos não funcionais são restrições aos serviços ou funções oferecidos pelo sistema. Incluem restrições de timing, restrições no processo de desenvolvimento e restrições impostas pelas normas. Ao contrário das características individuais ou serviços do sistema, os requisitos não funcionais, muitas vezes, aplicam-se ao sistema como um todo.

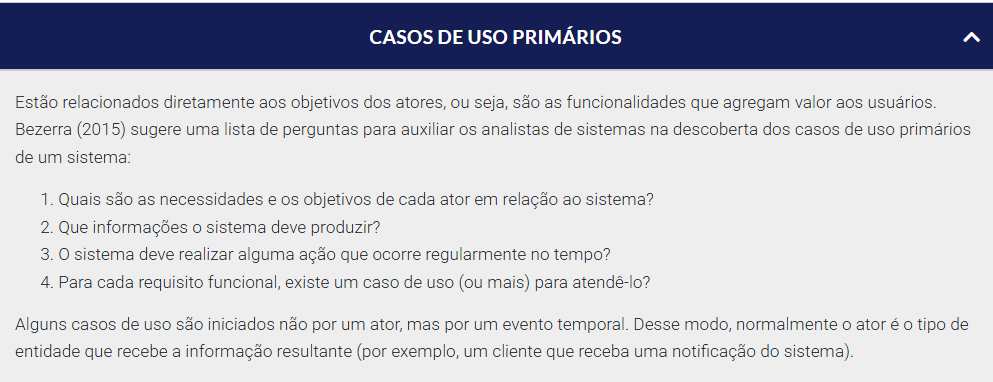
**Diagrama de Casos de Uso**

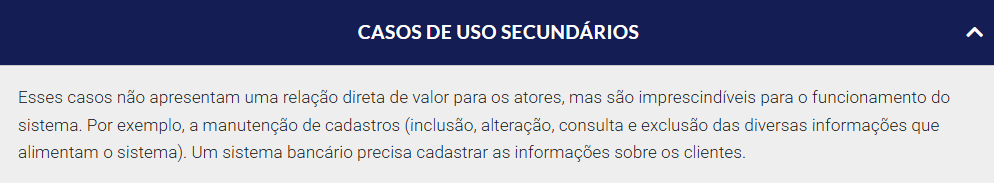
A visão de casos de uso da UML apresenta o conjunto de funções (ou usos) que o sistema deve executar para atender às necessidades dos usuários. No modelo de casos de uso, o sistema é visto sob a perspectiva do usuário.

O objetivo do Diagrama de Casos de Uso é apresentar uma visão geral dos requisitos funcionais do sistema e, portanto, ele é o digrama mais abstrato, flexível e simples da UML.

Caso de Uso, Atores e Relacionamentos

A forma mais comum e obrigatória dos relacionamentos é entre atores e casos de uso, que assume a representação gráfica de segmento de reta (uma linha cheia) ligando o ator ao Caso de Uso no qual ele interage.

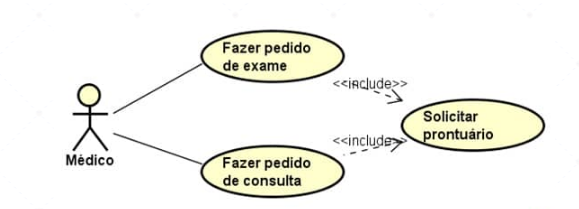




Um erro comum na elaboração de diagramas de casos de uso é estabelecer relacionamentos de comunicação entre casos de uso. Não existe troca de informações entre casos de uso, uma vez que são funcionalidades independentes entre si.

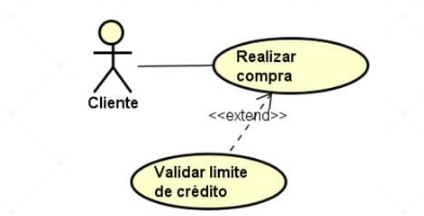
**Relacionamento de inclusão**

Tem como objetivo principal a reutilização. Suponha que uma parte dos passos de execução de um caso de uso se repita em diversos outros casos de uso. Podemos criar um caso de uso separado e “incluí-lo” nos demais. A inclusão funciona de forma semelhante à chamada de uma função em um programa.



**Relacionamento de extensão**

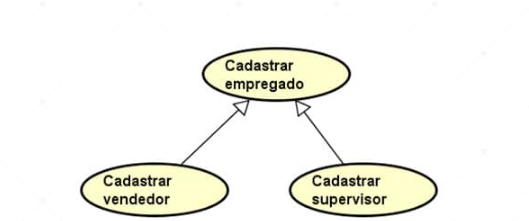
Tem como objetivo representar situações em que diferentes sequências de passos possam ser inseridas no mesmo caso de uso. Ou seja, dependendo de uma condição, o caso de uso segue um caminho diferente. Desse modo, um caso de uso estende outro, permitindo um comportamento eventual. Note que a existência do caso de uso estendido deve ser independente da existência de quaisquer casos de uso que o estendam.



**Relacionamento de generalização**

Pode existir entre dois casos de uso ou entre dois atores (conforme vimos em Ator). Esse relacionamento permite que um caso de uso (mais específico) herde características de outro (mais genérico).

Na generalização entre casos de uso, quando um caso de uso herda de outro, significa que as sequências de passos de execução do caso de uso mais genérico (pai) valem também para o mais específico (filho). Além disso, o caso de uso herdeiro participa de qualquer associação do qual o caso de uso mais genérico participe. Isso significa que um ator que interage com o caso de uso pai pode também interagir em qualquer caso de uso filho.



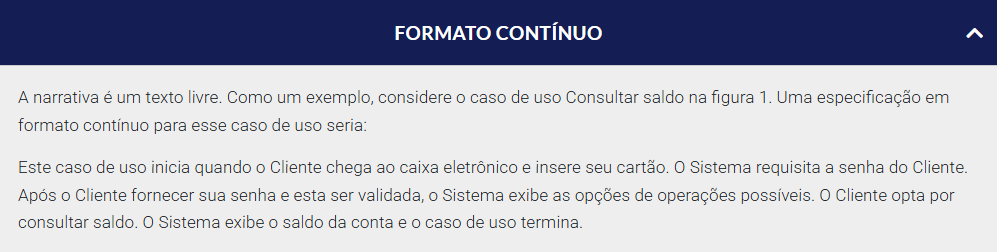
Módulo 2

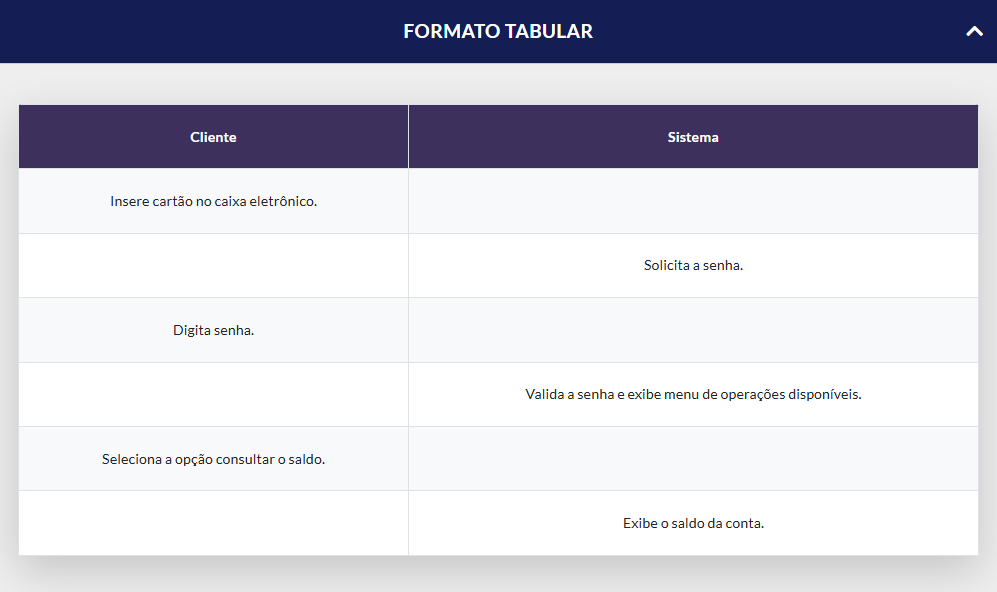
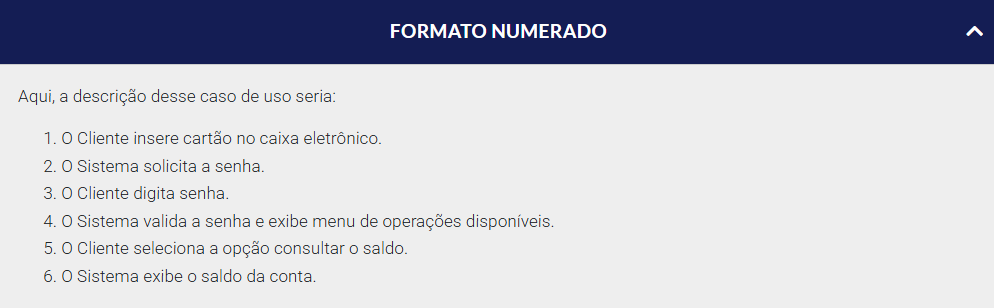
**Especificação e formatos de casos de uso**

Os casos de uso incluídos pelos analistas no Diagrama de Casos de Uso devem ser especificados em um formato de narrativa textual, para complementar a representação gráfica sob forma de diagramas. Desse modo, o Modelo de Casos de Uso é formado pelo diagrama e pelo conjunto de todas as descrições dos casos de uso.

O texto de um caso de uso é uma história que conta as ações ou a sequência de passos de um ou mais atores e do sistema para atingir determinado objetivo. Essa narrativa pode ser escrita em um parágrafo contínuo, em passos numerados ou em uma tabela contendo uma coluna para cada ator e uma coluna para o sistema.

A descrição de um caso de uso deve ser clara, livre de jargões e termos técnicos, portanto, de fácil entendimento não só para a equipe de desenvolvimento, mas para os usuários finais, pois o modelo de casos de uso é lido e validado por todos os envolvidos.

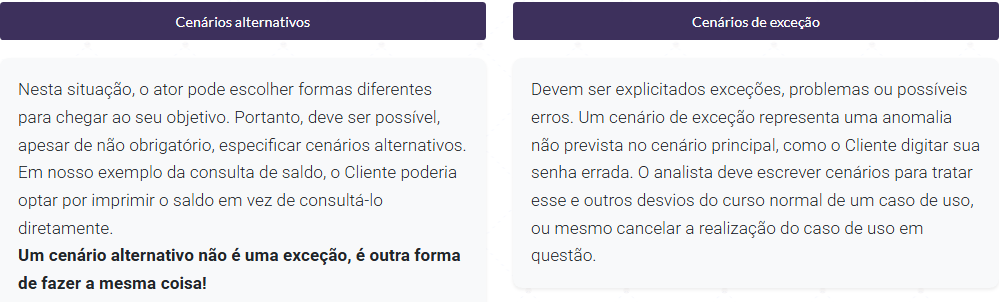


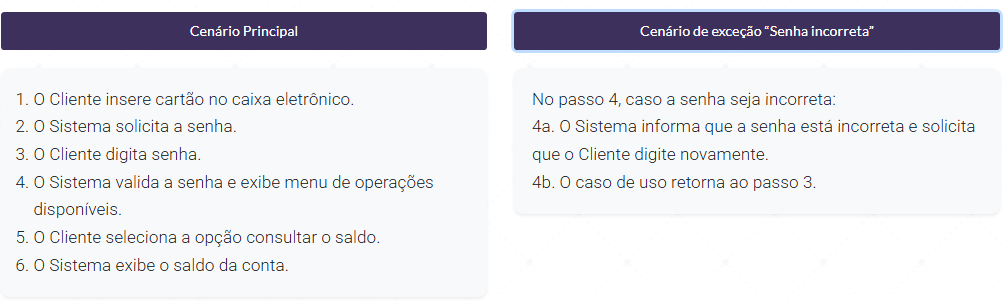


**Cenário principal e cenários alternativos**

Cada caso de uso possui apenas um cenário principal e ele deve ser descrito de forma clara e concisa, sem uso de jargão computacional. Não se esqueça: casos de uso devem ser escritos do ponto de vista do usuário e, sempre que possível, com os termos que os usuários possam compreender.

A forma correta de tratar exceções no modelo de casos de uso é especificar cenários de exceção que, diferentemente dos cenários alternativos, representam situações de anomalias que rompem o fluxo do cenário principal.





**Estrutura de especificação de casos de uso**

1. Identificador: código único para cada caso de uso que permite fazer a rastreabilidade com outros elementos e modelos. Por exemplo, com casos de teste e regras de negócio, UC01.

2. Nome: cada caso de uso tem um nome único que o identifica; é o nome que aparece no Diagrama de Casos de Usos. O nome do caso de uso é uma frase verbal iniciada por verbo no infinitivo. É importante ter cuidado com coerência e clareza ao nomear um caso de uso. Por exemplo, o nome do caso de uso UC01 é “Criar conta corrente”.

3. Objetivo: declaração sucinta do objetivo do ator no caso de uso. Para o caso de uso Criar conta corrente, o Objetivo seria: “O objetivo deste caso de uso é permitir que o Cliente do banco crie uma nova conta corrente”.

4. Ator primário: nome do ator que inicia o caso de uso ou ator que recebe o resultado do caso de uso; um caso de uso possui apenas um ator primário. Para o caso de uso Criar conta corrente, o Ator primário seria: “Cliente”.

5. Atores secundários: nomes dos demais atores que atuam no caso de uso. Para o caso de uso Criar conta corrente, um ator secundário seria: “Gerente da agência”.

6. Pré-condições: condições assumidas como verdadeiras para que o caso de uso tenha início; um caso de uso pode conter zero ou mais pré-condições. Por exemplo, em um caso de uso Criar conta corrente, uma pré-condição poderia ser “Cliente cadastrado com CPF válido”.

7. Cenário (ou fluxo) principal: sequência de passos que normalmente acontece quando o caso de uso é utilizado com sucesso; toda descrição de caso de uso deve ter um fluxo principal.

8. Cenários (ou fluxos) alternativos e de exceção: sequência de passos que apresentam formas de execução diferentes da descrita no fluxo principal ou situações de exceções e erros que devem ser tratadas pelo sistema. Não há obrigatoriedade de existirem fluxos alternativos ou de exceção em um caso de uso.

9. Pós-condições: estado do sistema após o caso de uso ter sido executado; tipicamente, uma pós-condição descreve que uma informação foi alterada, excluída ou inserida no sistema. Uma boa prática é usar verbo no particípio passado para descrever pós-condições. Por exemplo, em um caso de uso Criar conta corrente, uma pós-condição seria “conta corrente criada”.

10. Regras de negócio: a descrição de um caso de uso pode fazer referência a uma ou mais regras de negócio. Aprenda mais sobre Regras de Negócio em Explore +.

**Especificação de casos de uso com inclusão e extensão**

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Módulo 3

**Conceito e elementos de uma Classe**

A abordagem de desenvolvimento de sistemas orientada a objetos assume que um sistema é uma coleção de entidades – os objetos – que se comunicam e realizam ações. A interação entre objetos é a forma de execução de uma funcionalidade do software.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Uma classe pode ser definida como uma descrição de atributos (características) e serviços (ações) comuns a um grupo de objetos.

Tabela

O conteúdo gerado por IA pode estar incorreto.

**Visão geral do Diagrama de Classes**

O Diagrama de Classes da UML é o modelo em que representamos as classes de um sistema, incluindo os detalhes sobre seus atributos e suas operações, e como essas classes se relacionam.

Portanto, a elaboração do Diagrama de Classes é fundamental em todo o processo de desenvolvimento que segue a abordagem da orientação a objetos.

É comum chamarmos esse diagrama que evolui sucessivamente de modelo de classes de análise, modelo de classes de projeto e modelo de classes de implementação, fazendo referência às etapas do processo de desenvolvimento.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

Mas qual é o significado dessas associações? Quando ligamos as classes Cliente e Pedido significa que, durante a execução do sistema, haverá a possibilidade de troca de mensagens entre objetos dessas classes.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

As associações possuem diversas características que podem ser representadas, por escolha do analista, para prover um melhor entendimento do modelo: nome, multiplicidades, tipo de participação, direção de leitura e papéis.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto

O conteúdo gerado por IA pode estar incorreto.

**Classes associativas**

São classes que estão ligadas a associações, em vez de estarem ligadas a outras classes. São também chamadas de classes de associação. Esse tipo de classe normalmente aparece quando duas ou mais classes estão associadas e seja necessário manter informações sobre a associação que existe entre elas.

É muito comum a necessidade de classes associativas em relacionamentos muitos para muitos, ou seja 0..\* 0..\*. Uma classe associativa é representada pela mesma notação utilizada para uma classe comum. A diferença é que aquela é ligada por uma linha tracejada a uma associação.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Autoassociação**

Tem como objetivo relacionar objetos da mesma classe, mas cada objeto tem um papel distinto nessa associação. Por exemplo, na figura 11, objetos da classe Empregado estão associados entre si, um empregado como supervisor, que supervisiona outros empregados.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Todo-parte**

Existe uma categoria especial de relacionamento que possui uma semântica (significado) particular: todo-parte.

Uma relação todo-parte entre dois objetos indica que um dos objetos está contido (é uma parte do) no outro, ou seja, o objeto do lado todo contém o do lado parte.

**Atenção**

A **agregação** e a **composição** são casos especiais da associação, logo, todas as características válidas para uma associação valem para agregações e composições. Agregações/composições são assimétricas no sentido que, se um objeto A é parte de um objeto B, o objeto B não pode ser parte do objeto A.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

* **Agregação**: Representa uma relação "todo-parte" mais fraca. O objeto parte **pode existir independentemente** do todo. No UML, é representada por um **losango vazio**.
* **Composição**: É uma relação "todo-parte" mais forte. O objeto parte **não pode existir sem** o todo, ou seja, ele é criado e destruído junto com o objeto principal. No UML, usa um **losango preenchido (preto)**.

Ou seja, na agregação, os objetos podem viver separados; na composição, a vida deles está diretamente ligada.

**Herança**

Outro relacionamento com semântica específica é o de herança, ou relacionamento de generalização/especialização. Esse tipo de relacionamento, diferente dos que você estudou, significa um relacionamento de generalidade ou especialidade entre as classes envolvidas.

Nesse tipo de relacionamento, as classes mais específicas herdam características das classes mais genéricas.

Módulo 4

**Visão geral do Diagrama de Objetos**

Além do Diagrama de Classes, a UML define um segundo tipo de diagrama na perspectiva estática e estrutural do sistema: o Diagrama de Objetos.

Assim como objetos são instâncias de classes, os Diagramas de objetos podem ser vistos como instâncias de Diagramas de Classes.

Um Diagrama de Objetos UML representa uma instância específica de um Diagrama de Classes em determinado momento. Portanto, do ponto de vista gráfico, ele é bastante parecido com o Diagrama de Classes.

Os valores dos atributos podem sofrer alterações e, nesse caso, dizemos que o objeto sofreu uma **mudança de estado**.

Em um Diagrama de Objetos, cada objeto é representado por um retângulo com dois compartimentos. O compartimento superior mostra a identificação do objeto na forma de texto sublinhado:

**:NomeClasse**

**nomeObjeto: NomeClasse**

Diagrama

O conteúdo gerado por IA pode estar incorreto.

 Os relacionamentos no Diagrama de Objetos mostram uma situação específica em que aqueles objetos se encontram, não as possíveis relações entre eles.

Texto

O conteúdo gerado por IA pode estar incorreto.

Os Diagramas de Objetos podem ser usados também para verificar se o sistema foi desenvolvido conforme os requisitos, e até mesmo para analisar se uma regra de negócio está sendo cumprida.

**Visão geral do Diagrama de Pacotes**

Os pacotes servem para agrupar diagramas, tais como classes, casos de uso, ou mesmo outros pacotes. Normalmente, os grupos de classes são associados com módulos de programação do sistema. Uma classe só pode aparecer uma vez no mesmo pacote, mas pode estar presente em vários pacotes diferentes.

Um pacote representa um grupo de elementos que pode se relacionar com outros pacotes por meio de uma relação de dependência.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

**Desdobramentos dos Diagramas no Pacote**

Os pacotes se relacionam com outros pacotes por meio de um vínculo de dependência.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário

O conteúdo gerado por IA pode estar incorreto.

**Dependência de acesso** (linha contínua com triângulo branco): Indica que um pacote pode acessar elementos públicos de outro pacote sem necessariamente importá-los.

**Dependência de importação** (linha pontilhada com triângulo preto): Indica que um pacote importa visibilidade dos elementos do outro pacote, permitindo o uso direto sem qualificação do nome do pacote.

Diagrama

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

No contexto da modelagem de casos de uso, os pacotes podem ser utilizados com diversos objetivos. Alguns desses objetivos são: estruturar o modelo de casos de uso com base nos grupos de usuários do sistema; definir a prioridade na qual os casos de uso serão desenvolvidos; e definir o grau de correlação entre os casos de uso.

O critério para criar os pacotes é subjetivo e depende da visão e das necessidades do projetista.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Texto

O conteúdo gerado por IA pode estar incorreto.

Exercícios:

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo, Email

O conteúdo gerado por IA pode estar incorreto.

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.