

INSTITUTO FEDERAL
Santa Catarina

ANÁLISE DE SISTEMAS

CURSO TÉCNICO EM

INFORMÁTICA PARA INTERNET



Créditos

Equipe IFSC - Santa Catarina

Professores

Juliano Lucas Gonçalves
Marcos André Pisching
Clélio Marcos Ferreira
Cátia dos Reis Machado

Equipe Designa Design Instrucional

Coordenação de Projeto

Cíntia Costa
Ester Konig

Coordenação de Design Instrucional

Emily Mercuri

Design Instrucional

Joyce Paola Mangrich

Design Gráfico

Ayrin Barboza

Revisão ortográfica

Priscila Verçosa

**Este trabalho está licenciado
sob CC BY-NC 4.0** 



Ficha catalográfica

Apresentação

Nesta Unidade Curricular, você conhecerá métodos e técnicas de modelagem de sistemas de informação, considerando que a análise de sistemas é fundamental para estudantes e profissionais de várias áreas, especialmente aqueles envolvidos em tecnologia da informação e engenharia de software. Afinal, ela ajuda a compreender os problemas que precisam ser resolvidos em um contexto empresarial ou organizacional, como identificar as necessidades dos usuários e as metas do sistema.

Além disso, é essencial para definir os requisitos do sistema de forma precisa, pois coletar, documentar e analisar requisitos é necessário para garantir que o sistema a ser desenvolvido atenda às expectativas dos usuários e às necessidades do negócio. Economicamente, uma análise precisa dos sistemas pode evitar retrabalhos e custos desnecessários durante o desenvolvimento do software. Ao identificar corretamente os requisitos desde o início, os desenvolvedores evitam alterações tardias e problemas de escopo. E tem mais! A análise de sistemas facilita a comunicação com clientes, usuários e outros *stakeholders*, ajudando a desenvolver habilidades interpessoais e garantindo que todos os envolvidos tenham uma compreensão clara do sistema a ser desenvolvido.

Assim, você verá que a análise de sistemas fornece informações valiosas para apoiar a tomada de decisão estratégica e a alocação de recursos, pois, ao compreender os sistemas existentes e as necessidades dos usuários, é possível identificar oportunidades de inovação e melhorias contínuas nos processos de negócio e nos sistemas de software.

Portanto, esse conhecimento é essencial para o seu sucesso profissional, pois fornecerá habilidades e informações necessárias para projetar, desenvolver e manter sistemas de software eficazes e eficientes em uma ampla gama de campos relacionados à tecnologia da informação.

Objetivo

- Conhecer métodos e técnicas de modelagem de sistemas de informação.

Sumário



1. A importância da análise no processo de desenvolvimento de sistemas	3
2. Levantamento de requisitos	5
3. Modelagem de diagramas por meio da UML	13
4. Diagrama de casos de uso	21
5. Modelagem conceitual	48
Finalizando	64
Referências	65

1. A importância da análise no processo de desenvolvimento de sistemas

Para começar os estudos sobre a análise de sistemas, conheça, a seguir, as principais causas de fracasso no desenvolvimento de sistemas e o ciclo de vida básico para o desenvolvimento.



Mas, afinal, o que é análise? A análise é uma etapa fundamental no processo de desenvolvimento de sistemas. Consiste em identificar e definir o que o sistema precisa fazer e como ele deve funcionar para atender às necessidades dos usuários. Para que isso dê certo, é necessário entender as necessidades do cliente para o software que será desenvolvido, isso significa identificar as funcionalidades essenciais para priorizar o desenvolvimento das atividades mais críticas do usuário.

Para auxiliar nesse processo, temos o ciclo de vida de desenvolvimento, que é composto por sete etapas, que são:

Ciclo de vida de desenvolvimento

Levantamento de requisitos

Primeiramente, precisamos conhecer os requisitos (funcionalidades) e as restrições que o software deve ter ou respeitar. Aqui, aplicamos técnicas como questionário, entrevista, casos de uso, análise de documentos e observação.



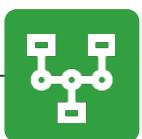
Projeto

Aqui, os requisitos descobertos anteriormente são transformados em software. Nesta fase, definimos a linguagem de programação a ser utilizada, o esquema de banco de dados e como deverão ser as interfaces.



Análise de requisitos

É o momento de propor a produção de um modelo que permita representar o problema a ser resolvido com, por exemplo, um diagrama de casos de uso.



Implementação

Agora, com base no projeto, podemos fazer a codificação ou programação do sistema, tendo o projeto como guia.



Teste

Nesse momento, devemos verificar se o que foi desenvolvido atende ou não a todas as especificações dos requisitos.



Implantação

Essa etapa consiste em instalar o software no ambiente do cliente e dar suporte inicial, quando necessário.



Manutenção

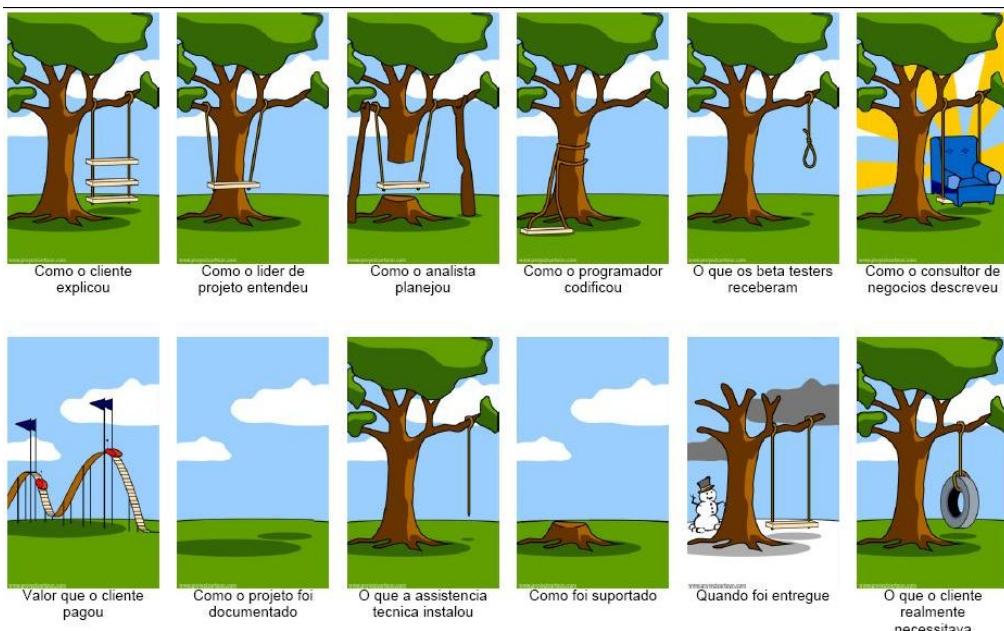
Essa fase serve para corrigir problemas ou erros que possam ocorrer na utilização do software.



Dependendo da metodologia de desenvolvimento a ser utilizada, a ordem das etapas vistas anteriormente pode ser alterada para melhor adaptação.

Agora, observe, na figura a seguir, a importância de realizar o ciclo de vida no processo de desenvolvimento.

Riscos no processo de desenvolvimento



Fonte: Uni System 51 (2016).

É importante ressaltar que a organização proposta pelo ciclo de vida tem por objetivo evitar que o software seja entregue de forma incorreta, gerando custos desnecessários ao cliente, o que contribuirá para sua insatisfação.

No próximo tópico, você conhecerá mais detalhes sobre a etapa de levantamento de requisitos, a primeira etapa do ciclo de vida.

2. Levantamento de requisitos

Quando **identificamos** os requisitos de um sistema, estamos essencialmente descobrindo quais são as coisas que o sistema precisa ser capaz de fazer (suas funcionalidades) e quais são as restrições ou limitações que estão associadas a essas funcionalidades.

Em outras palavras, estamos definindo o que o sistema deve fazer e quais são as condições ou regras que devem ser consideradas ao fazer essas coisas.

Uma exigência significa uma indicação do que o sistema deve realizar ou quais atributos ele deve possuir. Cabe destacar que as exigências são solicitações feitas pelo cliente, e não algo que a equipe concebe.

Durante um projeto de desenvolvimento de sistemas, serão elaboradas exigências descrevendo o seguinte (Dennis; Wixon; Roth, 2014).

Requisitos

01

As demandas do negócio
(requisitos do negócio).

02

O que os usuários
precisam realizar
(requisitos dos usuários).

03

O que o software deve
executar (requisitos
funcionais).

04

As características que o
sistema deve apresentar
(requisitos não funcionais).

05

Como o sistema deve ser
elaborado (requisitos do
sistema).

A seguir, conheça as características de cada uma dessas exigências.

Exigências do desenvolvimento do sistema

Exigências	Características
Regras de negócio	<p>As regras de negócio (RN) como um aspecto de uma empresa ou organização que se destina a descrever a estrutura ou o comportamento do negócio. Elas também podem ter características relacionadas ao controle de acesso. Por exemplo: para compras no cartão de crédito, o valor mínimo por parcela é de R\$ 60,00. Não serão aceitos pagamentos via Pix. O horário de funcionamento da loja é das 8h às 18h, sem fechar ao meio-dia.</p>
Requisitos do usuário	<p>Neste caso, eles descrevem as tarefas que os usuários realizam como uma parte integral das operações comerciais, como: "Agendar reunião com um cliente"; "Remeter novo pedido de um cliente"; "Reorganizar o estoque"; "Determinar o crédito disponível" e "Examinar os saldos das contas".</p>
Requisitos funcionais	<p>Os requisitos funcionais (RF) descrevem as funcionalidades do sistema, ou seja, o que o sistema deve fazer. Por exemplo: um sistema de e-commerce deve permitir ao usuário realizar compras, visualizar produtos, realizar pagamentos.</p>
Restrições lógicas	<p>As normas que regem as transações comerciais estão vinculadas às operações específicas. Por exemplo, ao realizar uma transação de venda, diversos critérios podem ser levados em conta, como a necessidade de aguardar a confirmação de pagamento pela operadora de cartão de crédito ou a impossibilidade de concluir a venda caso entregas anteriores ao mesmo endereço tenham sido devolvidas devido à inviabilidade do endereço (Wazlawick, 2014, p. 54).</p>

Restrições tecnológicas

As características da tecnologia utilizada para executar determinada função incluem interação com o usuário, tipo de protocolo de comunicação, requisitos de segurança, capacidade de tolerância a falhas etc. Por exemplo, determinar que a interface de usuário para fazer um pedido deve ter um padrão de design baseado em uma sequência de telas é uma restrição de interface que influencia a execução da função. Outro exemplo é a determinação de que "o processamento do pagamento não deve exceder 5 segundos". Trata-se de uma restrição de desempenho do sistema, que deve guiar as decisões do designer quanto ao método de comunicação com a operadora de cartão de crédito. Assim, o desenho do sistema deve contemplar uma conexão de banda larga dedicada às transações com as operadoras. (Wazlawick, 2014, p. 54)

Requisitos do sistema

Os requisitos do sistema (RS) se concentram na descrição de como criar o produto de software que será produzido pelo projeto.

Fonte: Elaborado pelos autores (2024).

Portanto, o levantamento de requisitos é um processo essencial que garante a compreensão completa das necessidades do cliente em um projeto de desenvolvimento de software. Em seguida, entram as técnicas de elicitação de requisitos, conforme você verá a seguir.

Técnicas de elicitação de requisitos

As técnicas de elicitação de requisitos dizem respeito ao “processo para descobrir os requisitos para um sistema de software pretendido, comunicando-se com o cliente, usuários finais, usuários do sistema e outros que tenham participação no desenvolvimento do sistema de software” (Schach, 2009, p. 34). Confira, a seguir, outros detalhes.

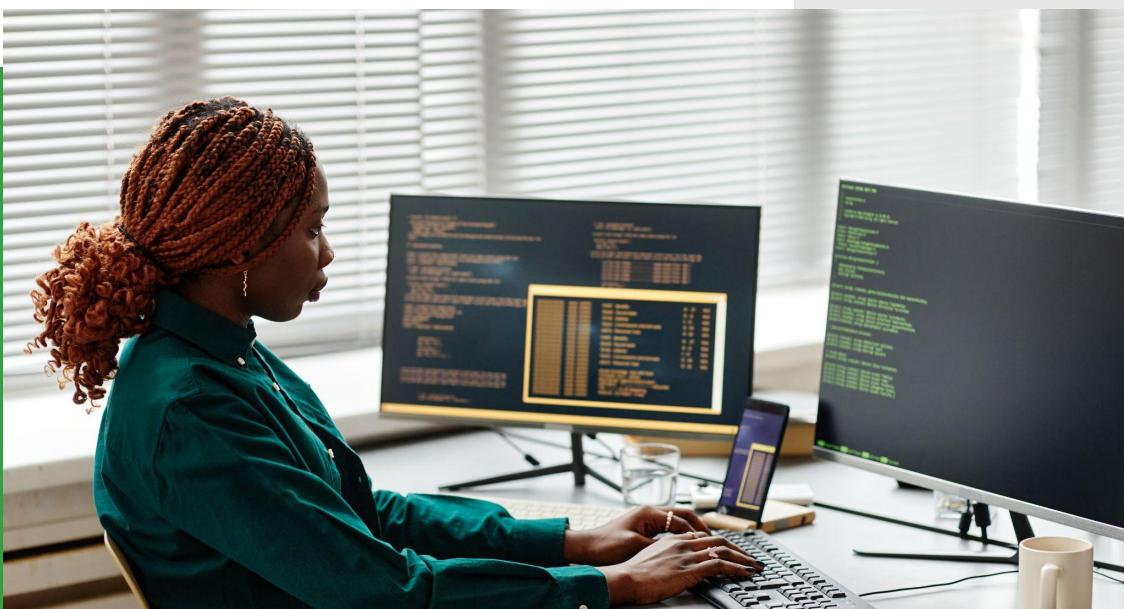
Técnicas de elicitação de Requisitos

Técnicas	Características
Entrevistas	As entrevistas são um meio forte para cobrar os requisitos. A organização pode realizar vários tipos de entrevistas, tais como: entrevistas estruturadas (fechadas), em que cada informação é única para coletar dados, sendo decidida com antecedência, porque elas seguem um padrão e a questão de discussão firmemente; entrevistas não estruturadas (abertas), em que a informação a reunir não é decidida com antecedência, mais flexível e menos tendenciosa; entrevistas orais; entrevistas escritas; entrevistas individuais que são realizadas entre duas pessoas em todo o país.
Pesquisa	A organização pode realizar pesquisas entre várias partes interessadas, que consultarão sobre as expectativas e os requisitos do próximo sistema.
Questionário	O documento, com um conjunto predefinido de questões objetivas, é entregue a todas as partes interessadas para ser respondido, assim, posteriormente, serão coletadas e compiladas as respostas. Uma lacuna dessa técnica é que, caso uma opção para algum problema não seja mencionada no questionário, a questão pode ser deixada sem supervisão.

Análise de tarefas	A equipe de engenheiros e desenvolvedores pode analisar a necessidade da operação do novo sistema. Se o cliente já tiver algum software para executar determinada operação, ele é estudado, e os requisitos do sistema proposto são coletados.
Brainstorm	O <i>brainstorming</i> ou tempestade de ideias, mais do que uma técnica de dinâmica de grupo, é uma atividade desenvolvida para explorar a potencialidade criativa de um indivíduo ou de um grupo, colocando-a a serviço de objetivos predeterminados.
Debate	Um debate informal é realizado entre as várias partes interessadas, e todos os seus insumos são registrados para análise adicional de requisitos.
Prototipagem	É a construção de uma interface de usuário, sem adicionar funcionalidades detalhadas para que o usuário interprete os recursos do produto de software pretendido. Isso ajuda a dar uma melhor ideia a respeito dos requisitos. Se não houver nenhum software instalado no trabalho final do cliente, para a referência do desenvolvedor, e o cliente não estiver ciente de seus próprios requisitos, o desenvolvedor criará um protótipo com base em requisitos inicialmente mencionados. O protótipo é mostrado ao cliente, e o feedback é anotado. Ele serve como uma entrada para a coleta de requisitos.
Observação	A equipe de especialistas visita a organização ou o local de trabalho do cliente. Eles observam o funcionamento real dos sistemas existentes, também analisam o fluxo de trabalho final do cliente e como os problemas de execução são tratados. A equipe em si desenha algumas conclusões que ajudam a formar os requisitos esperados do software.

Fonte: Schach (2009, p. 34).

Cabe ressaltar que todas as técnicas apresentadas anteriormente podem ser aplicadas no mesmo projeto, e lembre-se de que toda e qualquer conversa entre o analista e cliente deve ser registrada ou documentada. As conversas e os pedidos realizados com as equipes de desenvolvimento também devem ser documentados.



Deve ficar claro, também, que requisitos são algo que o cliente solicita, não algo que a equipe projeta. Um requisito é simplesmente uma declaração do que o sistema deve fazer ou de quais características ele precisa ter.

Durante um projeto de desenvolvimento de sistemas, serão criados requisitos descrevendo (Dennis; Wixo; Roth, 2014):

- **Requisitos do negócio:** as necessidades do negócio.
- **Requisitos dos usuários:** o que os usuários precisam fazer.
- **Requisitos funcionais:** o que o software deve fazer.
- **Requisitos não funcionais:** características que o sistema deve ter.
- **Requisitos do sistema:** como o sistema deve ser construído.

Acompanhe, agora, como funciona a modelagem de diagramas por meio da UML.

3. Modelagem de diagramas por meio da UML

A Linguagem de Modelagem Unificada (UML) surgiu com o objetivo de juntar as diversas linguagens de modelagem da época. A primeira versão da UML foi lançada em 1996, graças aos esforços de Booch, Jacobson e Rumbaugh.



Em 1997 foi adotada pelo Object Management Group (OMG), que é uma Organização Internacional que aprova padrões abertos para aplicações orientadas a objetos. Atualmente, encontra-se na versão 2.5.1 e tem 14 diagramas, e a documentação oficial pode ser consultada no site da OMG.

[Acesse](#)

Os arquitetos, engenheiros e desenvolvedores de software podem representar as informações por meio de uma notação gráfica para visualização, especificação, construção e documentação de artefatos de sistemas simples ou complexos de software.

A padronização proposta pela UML auxilia na definição de arquitetura de projetos de sistemas, incluindo tanto aspectos conceituais quanto concretos. É possível representar os processos de negócios e as funções do sistema. Além de itens concretos, como as classes escritas em determinada linguagem de programação, esquemas de bancos de dados e componentes de software reutilizáveis (Booch; Rumbaugh; Jacobson, 2006).

A aplicação prática da UML ocorre por meio de um conjunto de diagramas, cada um com um objetivo definido e com uma notação específica, a partir da qual todos os envolvidos no processo de desenvolvimento têm o mesmo entendimento sobre o

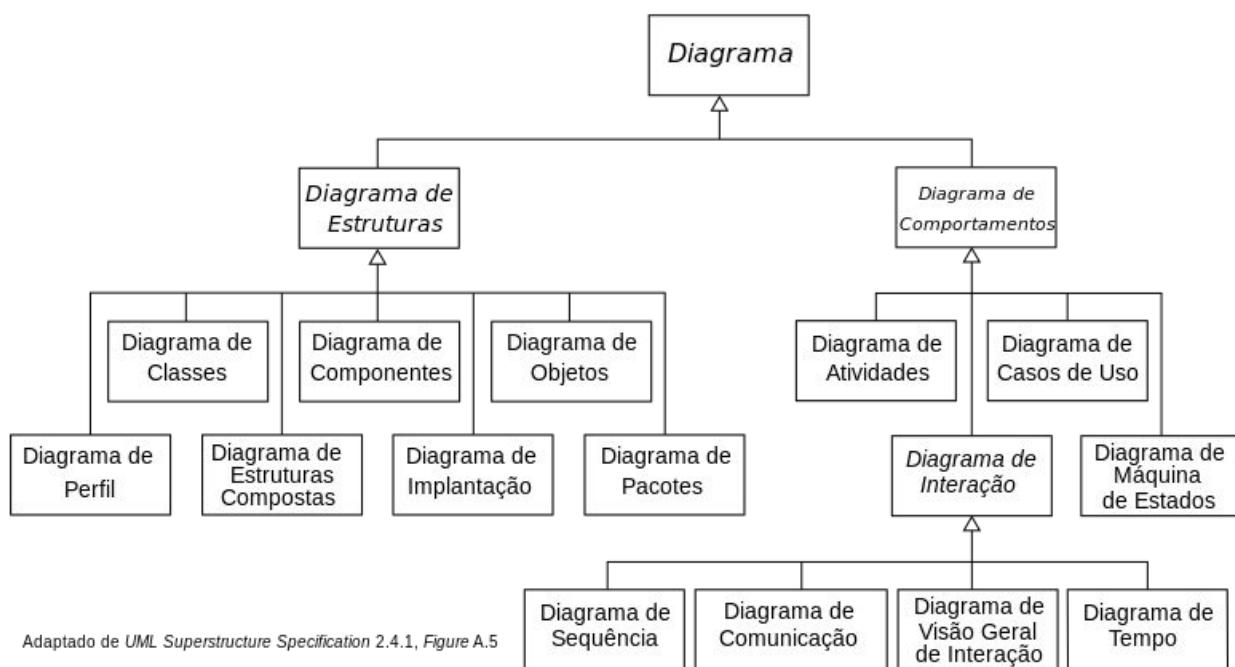
conhecimento que está sendo representado. Isso significa dizer que cada diagrama é utilizado para modelar e documentar as informações nas diferentes fases de desenvolvimento de softwares orientados a objetos. Além disso, pode ser utilizado independentemente da metodologia ou tecnologia escolhida para o desenvolvimento.



Diagramas da UML

Conforme a estrutura a seguir, podemos observar que os diagramas estão agrupados em **estruturais** e **comportamentais**.

Exigências do desenvolvimento do sistema



Fonte: Adaptado de OMG (2009).

Os **diagramas estruturais** são utilizados para definir o que deve ser implementado no sistema em termos de componentes, e ainda, a especificação da arquitetura do sistema. Nesse grupo, estão os diagramas de pacotes, classes, objetos, estrutura composta, componentes, perfil e implantação. O quadro a seguir apresenta a função de cada um deles:

Diagramas estruturais

Diagrama	Função
Diagrama de classe	Define a estrutura das classes presentes no sistema, determinando os atributos e as operações (métodos) de cada classe, além de estabelecer como as classes se relacionam e trocam informações. Esse tipo de representação pode incluir, ainda, definições de interfaces.
Diagrama de pacote	Utilizado para representar agrupamentos lógicos dentro do sistema. Pode ser utilizado de forma independente ou associado a outros diagramas.
Diagrama de objeto	Fornece uma visão dos valores armazenados pelos objetos em um determinado momento. Esse diagrama está associado ao diagrama de classes.
Diagrama de estrutura composta	Descreve a estrutura interna de uma classe, detalha as partes que a compõem, como se comunicam e colaboram entre si. Utilizado para descrever uma colaboração em que um conjunto de instâncias colaboram para realizar uma tarefa.
Diagrama de componentes	Representa os componentes do sistema quando ele for implementado e possíveis dependências entre tais elementos.

Diagrama de perfil	Possibilita estender os diagramas existentes, incluindo estruturas customizadas para uma determinada necessidade. Novos elementos podem ser criados.
Diagrama de implantação	Determina as necessidades de aparelhos físicos que serão necessários para executar o sistema. É, ainda, utilizado para representar como será a distribuição dos módulos do sistema.

Fonte: Adaptado de Guedes (2018).

Já os **diagramas comportamentais** são utilizados para representar o que deve acontecer no sistema ou processo de negócio. São usados para descrever as funcionalidades do sistema. Nesse grupo, estão diagramas de casos de uso, atividades e máquinas de estado. Confira a função de cada um deles no quadro a seguir.

Diagramas comportamentais

Diagrama	Função
Diagrama de casos de uso	Representa as funcionalidades (caso de uso) e as características de um sistema, assim como de que forma tais elementos se relacionam com usuários ou outras entidades externas (atores). Apresenta uma ideia geral de como o sistema deve se comportar. Normalmente, é utilizado em todo o processo de modelagem principalmente nas fases de levantamento e análise de requisitos do sistema. Serve de base para a criação de outros diagramas.
Diagrama de atividade	Descreve as atividades a serem executadas para a conclusão de um objetivo específico. É possível representar o fluxo de controle para a realização de uma atividade.

Diagrama de máquina de estado

Demonstra o comportamento de um elemento por meio dos diferentes estados que ele pode assumir. Como acontece a transição de um estado para outro.

Fonte: Adaptado de Guedes (2018).

Os **diagramas de interação** são um subconjunto dos diagramas comportamentais e descrevem os fluxos de controle entre diferentes componentes do sistema. Fazem parte deste grupo os diagramas de comunicação, sequência, tempo e visão geral de interação. No quadro a seguir, você pode conhecer a função de cada uma delas.

Diagramas comportamentais

Diagrama	Função
Diagrama de comunicação	Demonstra como os elementos estão vinculados e quais informações são trocadas no processo. Complementa o diagrama de sequência. Nesse diagrama, não há preocupação com a temporalidade do processo.
Diagrama de sequência	Apresenta a ordem temporal em que as mensagens são trocadas entre os objetos envolvidos em um determinado processo. Em geral, baseia-se nos casos de uso e utiliza o diagrama de classes para determinar os objetos das classes envolvidas em cada um dos processos. Identifica o evento gerador do processo, o ator responsável pelo evento, a sequência de troca de mensagens entre os objetos e os métodos chamados.
Diagrama de tempo	Descreve a mudança no estado ou condição de uma instância de uma classe ou seu papel em um determinado período. Ou seja, demonstra a mudança no estado de um objeto no tempo, respondendo a um evento externo.
Diagrama de visão geral de interação	Fornece uma visão geral de um sistema ou processo de negócio.

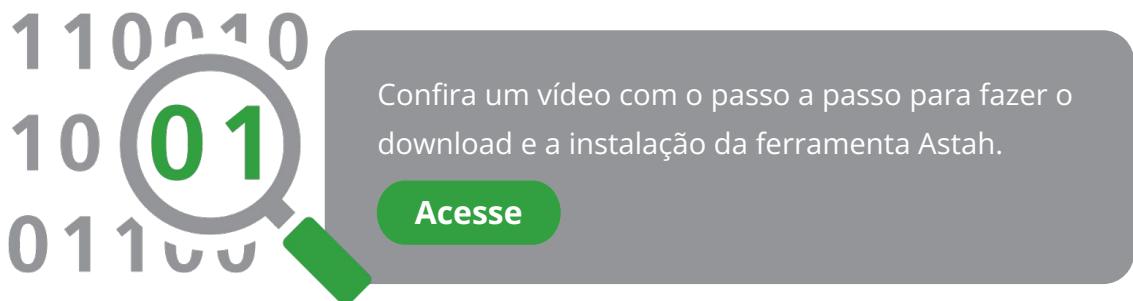
Fonte: Adaptado de Guedes (2018).

Cada diagrama possui um conjunto de elementos gráficos para representar as informações de acordo com sua função (Steinpichler; Kargl, 2011; OMG, 2017).

Além do conhecimento teórico e da notação gráfica, é necessário utilizar uma ferramenta para a construção dos diagramas. Existem várias opções disponíveis, entretanto, aqui apresentaremos a ferramenta Astah.

Astah

Essa é uma ferramenta de modelagem gráfica, intuitiva, em que é possível criar os diagramas propostos pela UML. Tem várias versões para download que podem ser escolhidas conforme a compatibilidade com o seu sistema operacional: Windows, Mac, Linux e iOS. Além disso, suporta algumas extensões para importação e exportação (PNG, EMF, SVG, JPEG, CSV, PDF, HTML, RTF e XML).

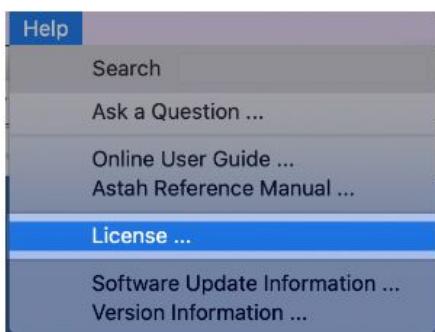


Para fazer o download da versão *free* para estudante do software, será necessário preencher seus dados com e-mail institucional. Depois, é só **clicar aqui** e seguir estes passos:

- 1** Fazer o download
- 2** Fazer a instalação
- 3** Verificar na sua caixa de e-mail e clicar no link para fazer o download de um arquivo xml
- 4** Após o download do arquivo, descompactá-lo
- 5** Abrir o Astah
- 6** No menu, clicar em [Help] - [License]

Observe, na imagem seguinte, como aparece o menu:

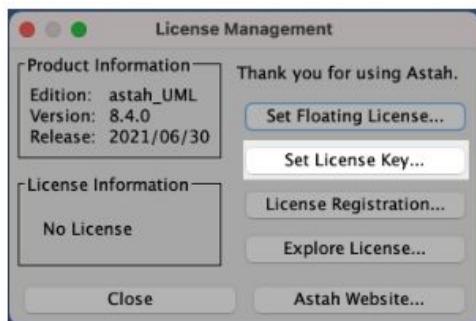
Menu



Fonte: Adaptado de Astah (2024).

Na janela da Licença, clique no botão [Set License Key] e selecione o arquivo da licença que foi baixado e descompactado (astah_uml_license.xml).

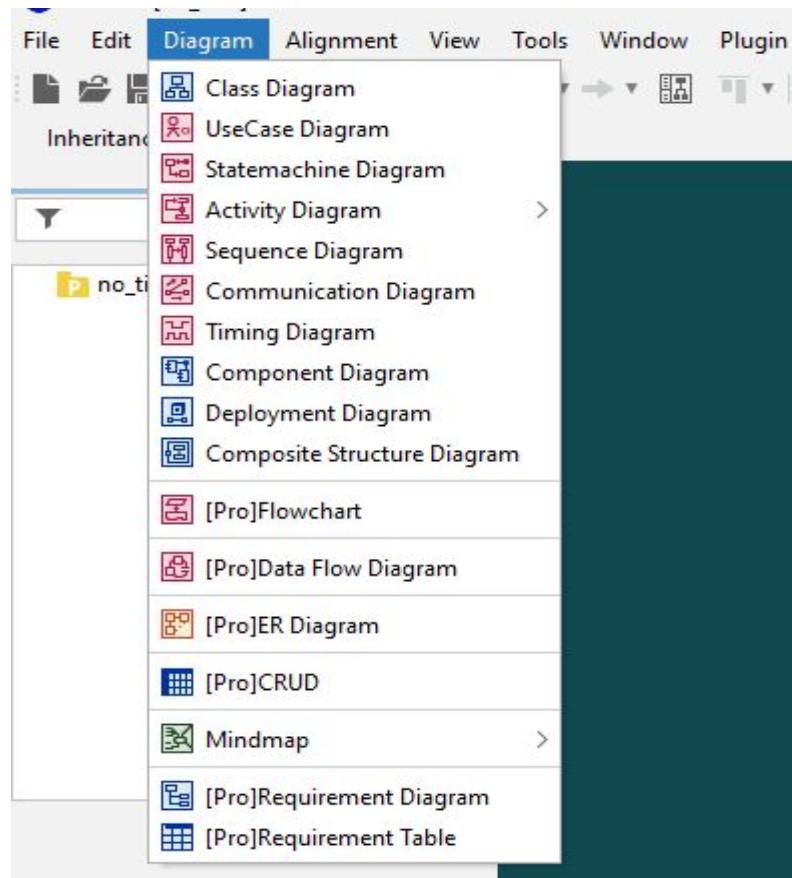
Licença



Fonte: Adaptado de Astah (2024).

Agora, a instalação está completa. Nesta versão, é possível criar os diagramas, conforme a figura a seguir.

Diagramas



Fonte: Adaptado de Astah (2024).

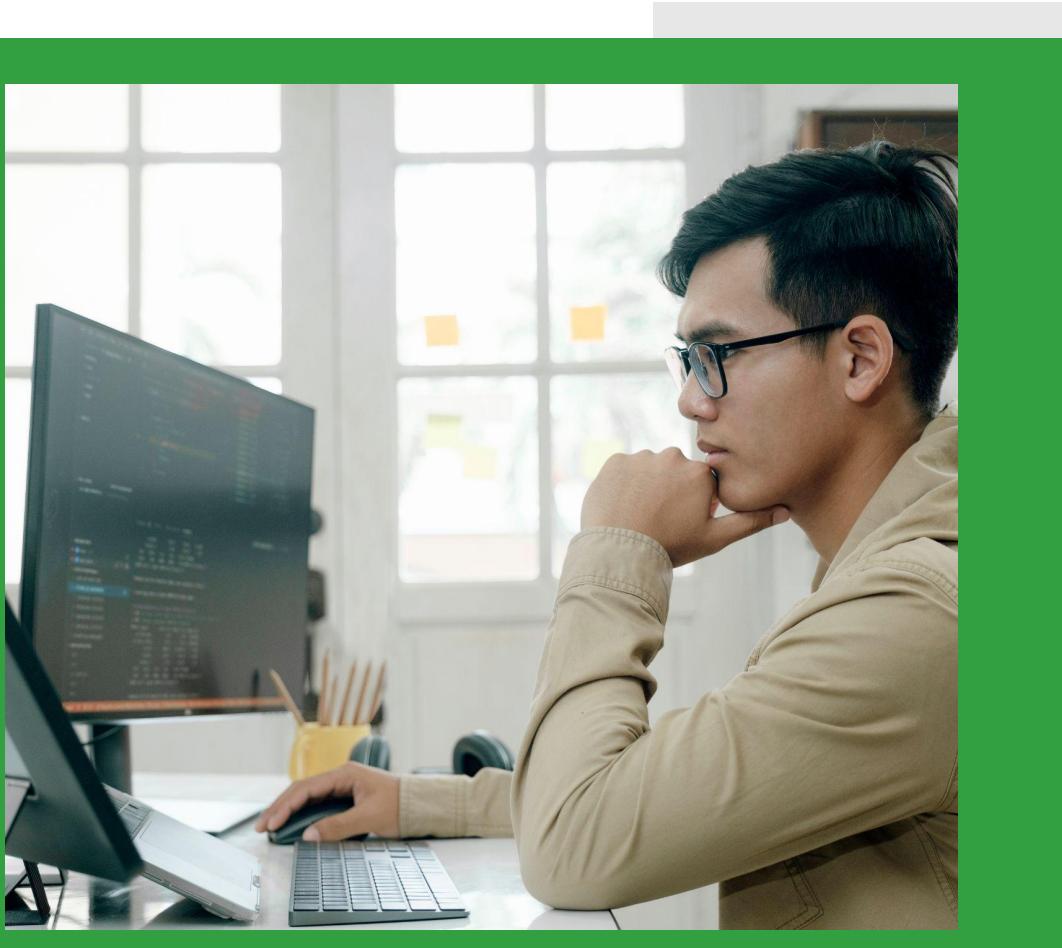
A partir de agora, é possível construir os diferentes tipos de diagrama. Vamos ver como funciona?

4. Diagrama de casos de uso

Nesta introdução ao diagrama de casos de uso em UML, você conhecerá as características desse diagrama, sua função, sua representação gráfica e outras informações importantes.

O que é um diagrama de casos de uso?

Um diagrama de casos de uso é uma representação visual dos requisitos funcionais de um sistema de software. Ele descreve as interações entre os atores externos (usuários, sistemas externos etc.) e o sistema em si, mostrando os diferentes casos de uso (ou funcionalidades) oferecidos pelo sistema. Esses casos de uso representam os diferentes caminhos que um usuário pode tomar para interagir com o sistema e atingir seus objetivos.



Ele é construído por meio de um processo interativo entre o cliente e os analistas, que conduzem a uma especificação do sistema no qual todos estão de acordo com o entendimento do projeto. Sintetizando, um caso de uso pode ser descrito como um documento narrativo que detalha a série de ações executadas por um ator ao interagir com um sistema para realizar um processo específico.

Para que serve um diagrama de casos de uso?

O diagrama de casos de uso serve como uma ferramenta de comunicação entre os *stakeholders* do projeto, ajudando a garantir uma compreensão comum dos requisitos do sistema. Ele também serve como uma base para o desenvolvimento do sistema, fornecendo uma visão geral das funcionalidades que o sistema deve oferecer e das interações entre os diferentes componentes. Além disso, o diagrama ajuda na identificação de requisitos incompletos, ambíguos ou conflitantes, permitindo que sejam resolvidos antes do desenvolvimento.

A narrativa do caso de uso deve enfatizar “o que” um processo faz e não “como” o processo é efetuado. Portanto, deve descrever os principais fluxos de ações do sistema e as sequências das possíveis exceções.

Existem algumas vantagens do uso de diagrama de casos de uso, veja quais são.

Vantagens do diagrama de casos de uso

1.

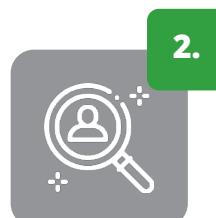
Comunicação clara



Permite uma comunicação eficaz entre desenvolvedores, designers, clientes e outros *stakeholders*, garantindo que todos tenham uma compreensão comum dos requisitos do sistema.

2.

Foco no usuário



Ajuda a manter o foco nas necessidades e nos objetivos dos usuários finais, garantindo que o sistema seja projetado para atender às suas necessidades.

3.

Identificação precoce de requisitos



Permite identificar e documentar os requisitos do sistema desde as fases iniciais do desenvolvimento, reduzindo o risco de retrabalho e atrasos no projeto.

Base para o design e desenvolvimento

Fornece uma base sólida para o design e desenvolvimento do sistema, ajudando a garantir que todas as funcionalidades necessárias sejam implementadas de forma eficiente.

4.



5.



Facilita a manutenção e evolução

Ao fornecer uma visão clara das funcionalidades do sistema e de suas interações, o diagrama facilita a manutenção e evolução do sistema ao longo do tempo.

E o que seria um ator nesse caso? Acompanhe a seguir!

Ator

Um ator, no contexto do diagrama de casos de uso em UML, representa um papel desempenhado por uma entidade externa ao sistema sendo modelado. Essa entidade pode ser um usuário humano, outro sistema, dispositivo de hardware ou até mesmo um objeto externo. Os atores interagem com o sistema por meio dos casos de uso, desempenhando papéis distintos na realização das funcionalidades do sistema.



Na notação UML, um ator é representado por um ícone de figura humana ou outro símbolo, posicionado fora do sistema, conectado por uma linha de associação aos casos de uso em que ele está envolvido. Geralmente, o nome do ator é colocado ao lado do símbolo para identificá-lo, veja a seguir um exemplo gráfico:

Representação gráfica do ator



Fonte: Elaborada pelos autores (2024).

Portanto, a representação gráfica de atores na notação UML desempenha um papel fundamental na visualização e compreensão das interações entre os usuários e os casos de uso em um sistema.

Veja, a seguir, alguns questionamentos que podem ajudar a identificar atores:

- Quem utilizará o sistema?
- Quem receberá informações do sistema?
- Quem fornecerá informações para o sistema?
- Quais outros sistemas utilizarão ou serão utilizados por este sistema?
- Em que organizações e áreas das organizações o sistema será utilizado?

Alguns exemplos de um ator em sistemas de e-commerce são:

Sistema de *e-commerce*



Cada um desses atores desempenha um papel específico dentro do sistema de *e-commerce*, interagindo com ele por meio dos casos de uso para realizar diferentes funcionalidades, como fazer compras, gerenciar estoque, processar pagamentos, entre outros.

Caso de Uso

Um caso de uso em UML representa uma interação entre um sistema e um ou mais atores (usuários ou sistemas externos) que descreve um conjunto de ações que o sistema executa para produzir um resultado observável de valor para um ator específico. Em outras palavras, um caso de uso descreve as funcionalidades do sistema do ponto de vista do usuário.



Na notação UML, um caso de uso é representado por um oval com o nome do caso de uso dentro dele. Os casos de uso são geralmente organizados dentro do limite do sistema, com linhas de associação conectando-os aos atores que estão envolvidos na interação. As linhas de associação indicam quais atores participam de cada caso de uso. Observe, a seguir, uma representação gráfica.

Representação gráfica dos casos de uso



Fonte: Elaborada pelos autores (2024).

Agora, confira alguns exemplos de um sistema de e-commerce:

Representação gráfica dos casos de uso



Realizar compra

Este caso de uso descreve o processo por meio do qual um cliente seleciona produtos, adiciona-os ao carrinho de compras e finaliza a compra, incluindo a seleção de método de pagamento e endereço de entrega.



Gerenciar estoque

Descreve como o sistema permite aos administradores adicionar, atualizar ou remover itens do estoque, além de visualizar informações sobre os níveis de estoque e alertas de reposição.



Visualizar produto

Este caso de uso descreve como os clientes podem navegar pelo catálogo de produtos, visualizar detalhes de um produto específico, incluindo imagens, descrição e preço.



Processar pagamento

Descreve o processo por meio do qual o sistema interage com o sistema de pagamento para processar a transação financeira após o cliente finalizar a compra.



Gerenciar conta de cliente

Permite aos clientes visualizar e editar informações de sua conta, como detalhes de contato, endereço de entrega e histórico de pedidos.

Cada um desses casos de uso descreve uma funcionalidade específica do sistema de *e-commerce*, mostrando como os diferentes atores interagem com o sistema para realizar tarefas relevantes.

Relacionamentos

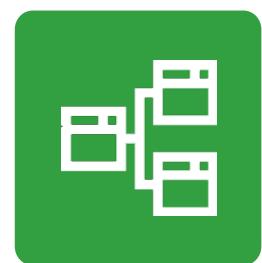
Um relacionamento nos casos de uso descreve como os diferentes casos de uso estão conectados ou interagem entre si. Existem três tipos de relacionamentos:



Associação



**Dependência:
Inclusão e
Extensão**



Generalização

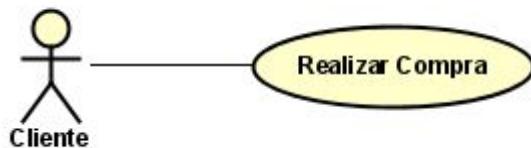
Esses relacionamentos ajudam a organizar e descrever as interações entre os diferentes casos de uso em um sistema de software, facilitando o entendimento de como as funcionalidades do sistema estão conectadas e como elas se comportam em conjunto.

A seguir, você vai conhecer a característica de cada um dos tipos.

Associação

A associação entre atores e casos de uso é essencial para definir quais atores estão envolvidos em quais as funcionalidades do sistema. Essa associação é representada por linhas de associação que ligam os atores aos casos de uso correspondentes no diagrama. Cada linha de associação indica que um ator está participando do caso de uso específico. Veja, na figura a seguir, um exemplo:

Exemplo de associação



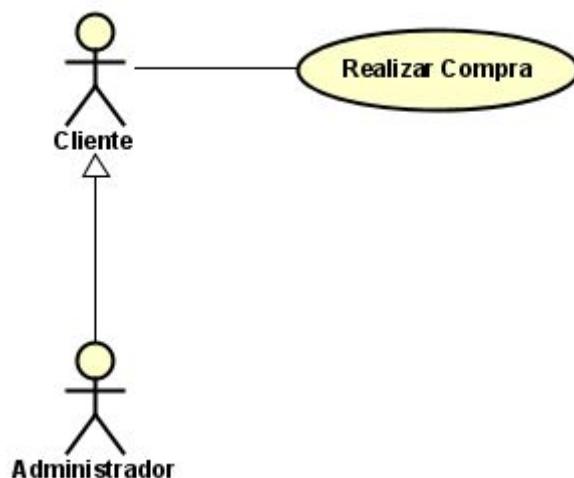
Fonte: Elaborada pelos autores (2024).

No diagrama de um sistema de *e-commerce*, a associação entre o ator "Cliente" e o caso de uso "Realizar Compra" indica que o cliente está envolvido na funcionalidade de realizar uma compra no sistema.

Generalização

A generalização de atores ocorre quando dois ou mais atores podem se comunicar com o mesmo conjunto de casos de uso. Um filho (herdeiro) pode se comunicar com todos os casos de uso que seu pai se comunica. Uma dica é colocar o herdeiro embaixo. Por exemplo:

Exemplo de generalização



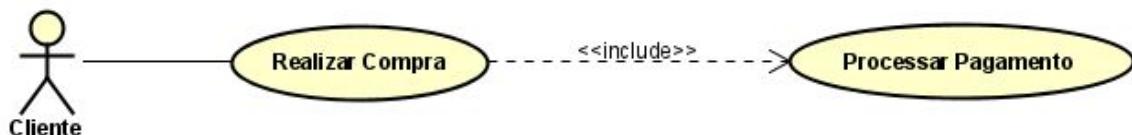
Fonte: Elaborada pelos autores (2024).

No diagrama de um sistema de *e-commerce*, a generalização ocorre entre o ator "Cliente" (pai) e o ator "Administrador" (filho). No caso, o "Administrador" poderá realizar as mesmas ações que o "Cliente", no caso de uso "Realizar Compra".

Dependência: inclusão (includes)

A inclusão é um relacionamento entre casos de uso em que um caso de uso (incluso) é totalmente executado dentro de outro caso de uso (includente). Isso é usado quando há um conjunto comum de ações que são compartilhadas por vários casos de uso. O caso de uso incluído não pode ser executado sozinho, ele sempre é chamado por outro caso de uso:

Exemplo de dependência: inclusão (includes)



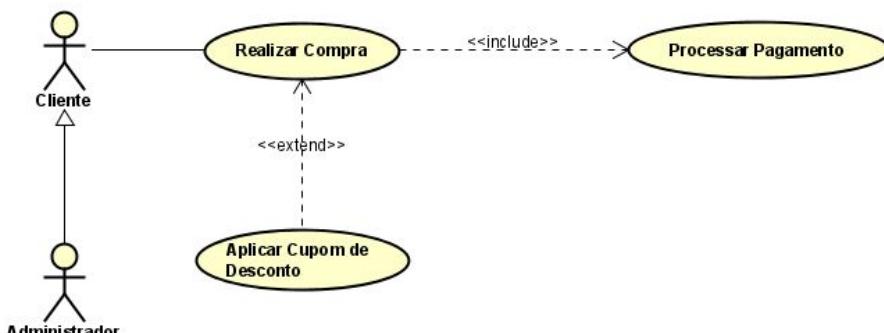
Fonte: Elaborada pelos autores (2024).

No sistema de *e-commerce*, o caso de uso "Realizar Compra" pode incluir o caso de uso "Processar Pagamento". Ou seja, o processo de realização de uma compra sempre inclui o processo de processar o pagamento.

Dependência: extensão (extends)

A extensão é outro relacionamento entre casos de uso, em que um caso de uso estende outro caso de uso com novos comportamentos opcionais. Isso é útil quando há funcionalidades que podem ser ativadas ou desativadas com base em certas condições durante a execução do caso de uso base. Por exemplo:

Dependência: extensão (extends)



Fonte: Elaborada pelos autores (2024).

No sistema de *e-commerce*, o caso de uso "Realizar Compra" pode estender o caso de uso "Aplicar Cupom de Desconto". Isso permite que os clientes apliquem um cupom de desconto durante o processo de compra, mas essa funcionalidade é opcional e só ocorre se o cliente possuir um cupom válido.

Etapas para criação de um diagrama de casos de uso

Para criar um diagrama de casos de uso, é preciso seguir algumas etapas, como você pode conferir a seguir.

Etapas	O que fazer
Identificar atores	<p>Análise de stakeholders: identifique todas as partes interessadas (<i>stakeholders</i>) que interagem com o sistema.</p> <p>Mapeamento de papéis: identifique os diferentes papéis que esses <i>stakeholders</i> desempenham no sistema.</p>
Identificar casos de uso	<p>Análise de requisitos: entenda os requisitos funcionais do sistema por meio de documentação, entrevistas com <i>stakeholders</i> etc.</p> <p>Identificação de funcionalidades: identifique as diferentes funcionalidades que o sistema deve fornecer para atender aos requisitos dos usuários.</p>
Estabelecer relacionamentos	<p>Associação de atores: associe os atores aos casos de uso com os quais eles interagem.</p> <p>Identificar inclusões e extensões: determine se há casos de uso que podem ser incluídos em outros ou estendidos por outros casos de uso.</p>

Desenhar diagrama	<p>Esboço inicial: comece com um esboço simples do diagrama, posicionando os atores e casos de uso e conectando-os com linhas de associação.</p> <p>Refinamento: refine o diagrama, adicionando detalhes como inclusões, extensões e nomes claros para os casos de uso e atores.</p>
Validar	<p>Valide o diagrama com <i>stakeholders</i> para garantir que todos os requisitos e interações do sistema estejam adequadamente representados.</p>

Fonte: Elaborada pelos autores (2024).

Vamos verificar como funciona na prática?

Exemplo de um sistema de *e-commerce*

Conheça, agora, o detalhamento dos casos de uso.

Detalhamento

Visualizar produto	Permite aos clientes navegarem pelo catálogo de produtos, visualizar detalhes de um produto específico, incluindo imagens, descrição e preço. Essa rotina permite a seleção de produtos para o carrinho de compra do cliente, desde que ele esteja logado em sua conta.
Realizar compra	Permite o cliente finalizar as compras que estão adicionadas no carrinho de compras, definindo a forma de pagamento o endereço de entrega. O caso de uso “aplicar cupom de desconto” pode estender a esse caso.

Processar pagamento

Esse processo interage com o ator “sistema de pagamento” para processar a transação financeira, sendo executado quando o cliente finaliza sua compra. O caso de uso “processar pagamento” deve ser incluído o caso de uso “realizar compra”, pois faz parte desse processo. Se for efetivado o pagamento, o caso de uso “entregar produto” pode estender o caso de uso “processar pagamento”, o qual está associado ao ator “sistema de entrega”.

Gerenciar conta de cliente

Permite aos clientes visualizar e editar informações de sua conta, como detalhes de contato, endereço de entrega e histórico de pedidos.

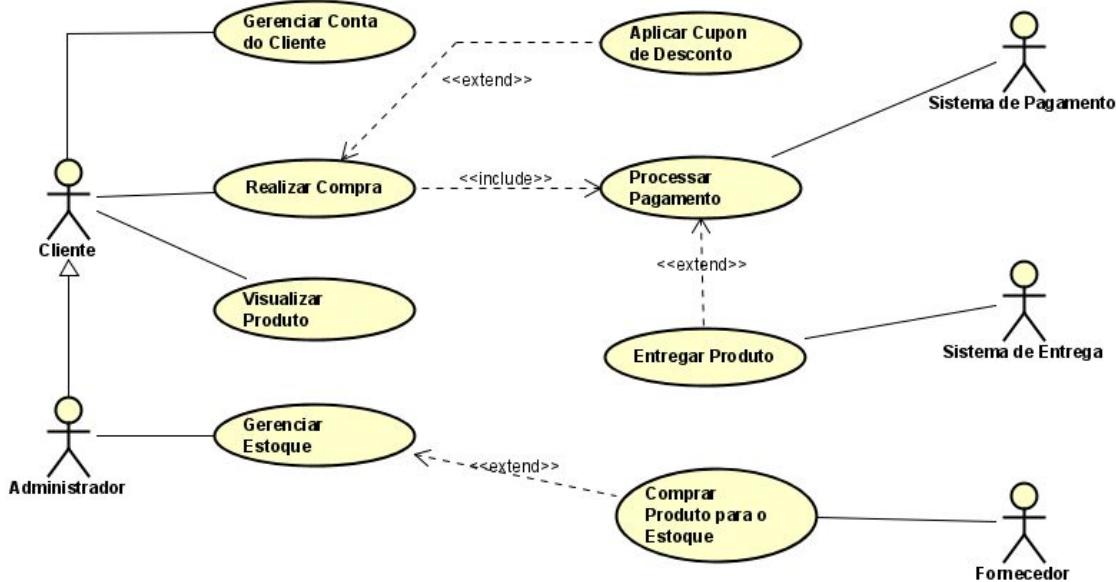
Gerenciar estoque

Permite a manutenção no estoque dos produtos que estão disponíveis no catálogo de vendas. O caso de uso “gerenciar estoque” está associado ao ator “administrador”, que é responsável por gerenciar o estoque do sistema. Caso necessite abastecer o estoque, o caso de uso “comprar produto para o estoque” pode estender o caso de uso “gerenciar estoque”, pois é uma parte do processo do gerenciamento, o qual está associado ao ator “fornecedor”.

Fonte: Elaborada pelos autores (2024).

O administrador tem permissão para realizar todas as funções do cliente. Observe o diagrama a seguir.

Diagrama de casos de uso: sistema de e-commerce



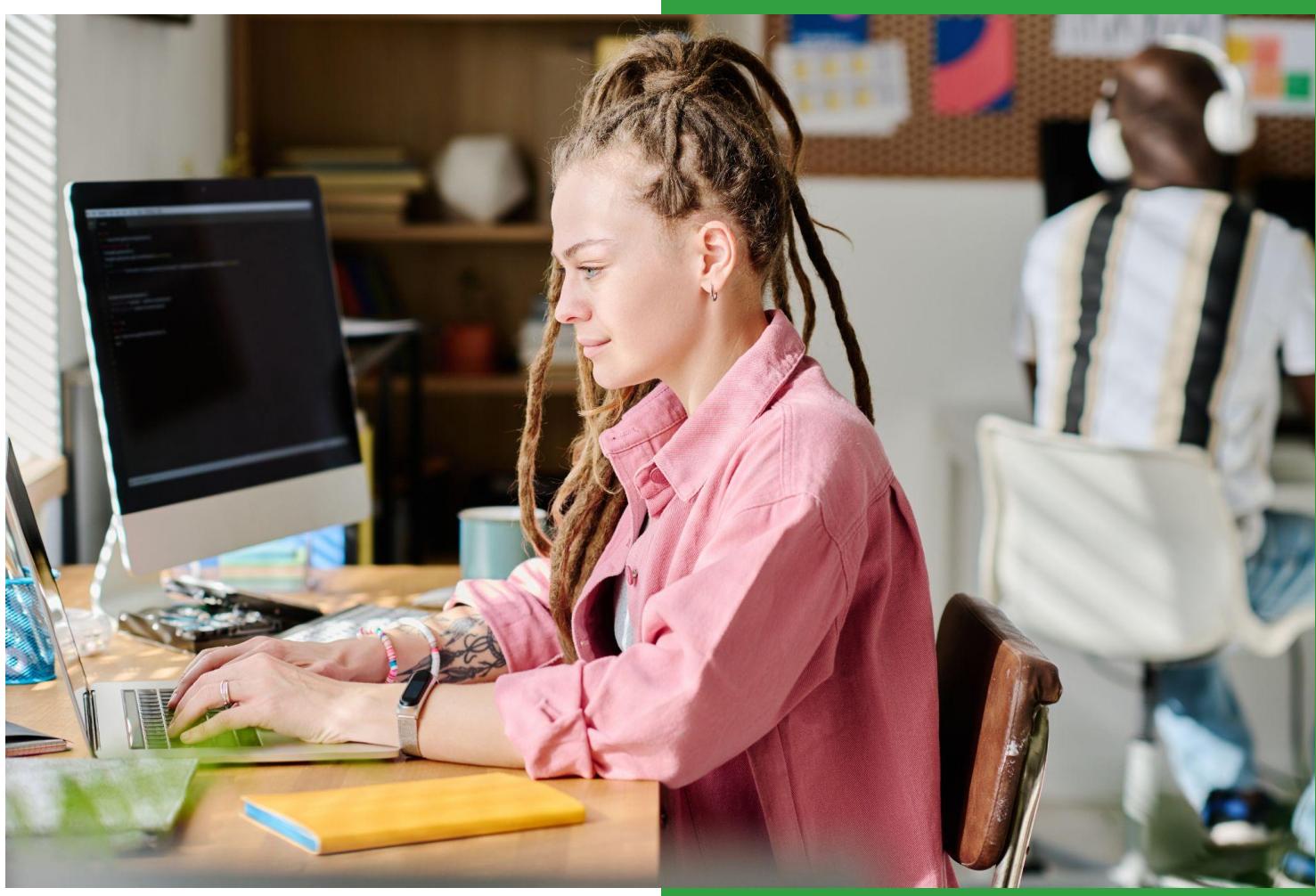
Fonte: Elaborada pelos autores (2024).

Este é um exemplo simplificado de um diagrama de casos de uso para um sistema de *e-commerce*, mostrando os atores envolvidos, os casos de uso e os relacionamentos entre eles. Esse diagrama serve como uma representação visual das funcionalidades principais do sistema.

Boas práticas e dicas

Na elaboração de diagramas, é fundamental adotar boas práticas e seguir algumas dicas essenciais.

Primeiramente, é importante manter a simplicidade, garantindo que o diagrama seja fácil de entender e interpretar. Além disso, é recomendável utilizar uma linguagem clara e consistente ao longo de todo o processo de documentação, evitando ambiguidades e garantindo uma comunicação eficaz.



Outro passo crucial é revisar e validar o diagrama com os *stakeholders* envolvidos, assegurando que ele atenda às expectativas e necessidades de todos os interessados.

Por fim, é imprescindível utilizar ferramentas apropriadas para desenhar e documentar o diagrama, garantindo precisão e eficiência na representação visual das informações. Essas práticas e dicas contribuem significativamente para a qualidade e utilidade dos diagramas produzidos.

Importância do diagrama de casos de uso na engenharia de software

O diagrama de casos de uso desempenha um papel crucial na engenharia de software por várias razões, como:

Diagrama de casos de uso na engenharia de software

1.

Comunicação clara



Fornece uma maneira eficaz de comunicar as funcionalidades do sistema entre *stakeholders*, incluindo clientes, desenvolvedores e gerentes de projeto.

2.

Identificação de requisitos

Ajuda a identificar e documentar os requisitos funcionais do sistema, garantindo que todas as partes interessadas tenham uma compreensão comum das funcionalidades esperadas.



3.

Análise de cenários



Permite a análise e validação de diferentes cenários de uso do sistema, ajudando a identificar possíveis lacunas ou inconsistências nos requisitos.

4.

Base para o design

Serve como base para o design detalhado do sistema, ajudando os desenvolvedores a entender como as diferentes partes do sistema se encaixam e interagem entre si.



5.

Documentação



Fornece uma documentação visual das funcionalidades do sistema, facilitando a manutenção e a compreensão do sistema ao longo do tempo.

Em resumo, o diagrama de casos de uso é uma ferramenta valiosa que ajuda a garantir o sucesso de projetos de software, fornecendo uma representação clara e concisa das funcionalidades do sistema e suas interações com os usuários e outros sistemas.

Descrição de casos de uso

A descrição de casos de uso é uma parte essencial do processo de modelagem de sistemas de software. Ela descreve em detalhes como os usuários interagem com o sistema e quais funcionalidades são oferecidas.

Existem três níveis de descrição de casos de uso: informal, típica e detalhada

A seguir, você vai conhecer as características de cada um dos níveis.

Descrição Informal

A descrição informal é uma visão geral e simplificada do caso de uso, geralmente usada nas fases iniciais do desenvolvimento de software para comunicar os conceitos-chave aos stakeholders de forma acessível. Essa descrição é escrita em linguagem natural e não segue uma estrutura rígida. Geralmente, inclui os elementos apresentados a seguir.



Elementos da descrição informal

01 Nome do caso de uso

Um título descritivo que identifica a funcionalidade em questão.

02 Objetivo

Uma breve declaração sobre o que o caso de uso realiza.

03 Pré-condições

Condições que devem ser verdadeiras antes que o caso de uso possa ser executado.

04 Fluxo básico

Uma descrição sequencial de passos que ocorrem normalmente durante a execução do caso de uso.

05 Fluxos alternativos

Descrição de cenários alternativos ou excepcionais que podem ocorrer durante a execução do caso de uso.

06 Pós-condições

Estado do sistema após a conclusão bem-sucedida do caso de uso.

Como exemplo de descrição informal do caso de uso, vamos pensar no seguinte caso: “realizar compra” de um sistema de *e-commerce*. Observe a seguir.

Exemplo de Descrição Informal

Nome do caso de uso	Realizar compra
Objetivo	Permitir que o cliente selecione produtos, adicione ao carrinho de compras e finalize a compra de maneira conveniente e eficiente.
Pré-condições	O cliente está autenticado no sistema. O cliente tem acesso à internet e a um dispositivo compatível. Os produtos desejados estão disponíveis para compra.

Fluxo básico	<p>O cliente navega pelo catálogo de produtos disponíveis.</p> <p>O cliente seleciona os produtos desejados e os adiciona ao carrinho de compras.</p> <p>O cliente revisa os itens no carrinho de compras e ajusta as quantidades, se necessário.</p> <p>O cliente avança para o processo de pagamento.</p> <p>O cliente seleciona o método de pagamento preferido e fornece as informações necessárias.</p> <p>O sistema valida as informações de pagamento e processa a transação.</p> <p>O cliente recebe uma confirmação da compra e um número de pedido.</p>
Fluxos alternativos	<p>Se o cliente não estiver autenticado, ele será redirecionado para o processo de login antes de continuar.</p> <p>Se ocorrer um erro durante o processo de pagamento, o cliente receberá uma mensagem de erro e deverá tentar novamente ou selecionar outro método de pagamento.</p>
Pós-condições	<p>Os produtos são reservados no estoque.</p> <p>O cliente recebe uma confirmação da compra por e-mail.</p> <p>O pedido é registrado no sistema para processamento e envio.</p>

Fonte: Elaborada pelos autores (2024).

Essa descrição informal fornece uma visão geral do caso de uso "realizar compra", destacando os passos principais envolvidos no processo de compra em um sistema de e-commerce.

Descrição típica

A descrição típica fornece mais detalhes do que a descrição informal, mas ainda mantém uma abordagem de alto nível. Ela é usada para documentar casos de uso durante as fases intermediárias do desenvolvimento de software, fornecendo informações mais estruturadas e detalhadas. Além dos elementos encontrados na descrição informal, a descrição típica também pode incluir:



Atores envolvidos

Lista dos atores que participam do caso de uso.



Diagrama de sequência

Diagrama que mostra a interação entre os atores e o sistema durante a execução do caso de uso.



Regras de negócio

Regras específicas que se aplicam ao caso de uso.

Agora, vamos a um exemplo de descrição típica do caso de uso também sobre “realizar compra” de um sistema de *e-commerce*:

Exemplo de Descrição Típica

Nome do caso de uso	Realizar compra
Objetivo	Permitir que o cliente selecione produtos, adicione ao carrinho de compras e finalize a compra de maneira conveniente e eficiente.
Atores envolvidos	Cliente, sistema de pagamento.
Pré-condições	<p>O cliente está autenticado no sistema.</p> <p>O cliente tem acesso à internet e um dispositivo compatível.</p> <p>Os produtos desejados estão disponíveis para compra.</p>
Fluxo básico	<p>O cliente navega pelo catálogo de produtos disponíveis.</p> <p>O cliente seleciona os produtos desejados e os adiciona ao carrinho de compras.</p> <p>O cliente revisa os itens no carrinho de compras e ajusta as quantidades, se necessário.</p> <p>O cliente avança para o processo de pagamento.</p> <p>O cliente seleciona o método de pagamento preferido e fornece as informações necessárias.</p> <p>O sistema valida as informações de pagamento e processa a transação.</p> <p>O cliente recebe uma confirmação da compra e um número de pedido.</p>

Fluxos Alternativos

Se o cliente não estiver autenticado, ele será redirecionado para o processo de login antes de continuar.

Se ocorrer um erro durante o processo de pagamento, o cliente receberá uma mensagem de erro e deverá tentar novamente ou selecionar outro método de pagamento.

Pós-condições

Os produtos são reservados no estoque.

O cliente recebe uma confirmação da compra por e-mail.

O pedido é registrado no sistema para processamento e envio.

Regras de negócio

O sistema não permite que o cliente finalize uma compra se o carrinho de compras estiver vazio.

O sistema aplica descontos ou promoções automaticamente, se disponíveis.

Diagrama de sequência

Um diagrama de sequência pode ser criado para visualizar a interação entre o cliente, o sistema de *e-commerce* e o sistema de pagamento durante o processo de realização da compra.

Fonte: Elaborada pelos autores (2024).

Essa descrição típica fornece uma visão mais estruturada do caso de uso “realizar compra”, incluindo detalhes sobre os atores envolvidos, pré-condições, fluxo básico, fluxos alternativos, regras de negócio e possíveis diagramas de sequência associados.



Descrição detalhada

A descrição detalhada é a mais abrangente e precisa conter todos os elementos da descrição informal e típica, sendo usada nas fases avançadas do desenvolvimento de software, quando os requisitos estão bem definidos e a implementação está próxima. Ela fornece uma descrição completa e precisa de todos os aspectos do caso de uso.

Aspectos da descrição detalhada

01 Cenários

Descrição detalhada de todos os cenários possíveis, incluindo fluxo básico, fluxos alternativos e excepcionais.

02 Interface do usuário

Detalhes sobre a interface do usuário envolvida no caso de uso.

03 Dados envolvidos

Descrição dos dados manipulados pelo caso de uso, incluindo estrutura, formato e regras de validação.

04 Requisitos não funcionais

Requisitos de desempenho, segurança, usabilidade etc.

05 Testes

Casos de teste associados ao caso de uso.

Veja um exemplo de descrição detalhada do caso de uso sobre “realizar compra” de um sistema de e-commerce.

Exemplo de Descrição Detalhada

Nome do caso de uso	Realizar compra
Objetivo	Permitir que o cliente selecione produtos, adicione ao carrinho de compras e finalize a compra de maneira conveniente e eficiente.
Atores envolvidos	Cliente, sistema de pagamento.
Pré-condições	O cliente está autenticado no sistema. O cliente tem acesso à internet e a um dispositivo compatível. Os produtos desejados estão disponíveis para compra.
Fluxo básico	<p>O cliente acessa a página inicial do sistema de e-commerce e navega pelo catálogo de produtos. O sistema exibe uma lista de produtos, incluindo imagens, descrições, preços e opções de compra. O cliente seleciona um produto e o adiciona ao carrinho de compras, especificando a quantidade desejada. O sistema atualiza o carrinho de compras, exibindo o item adicionado, juntamente com o subtotal e opções adicionais. O cliente continua navegando e adicionando mais produtos ao carrinho, repetindo os passos 3 e 4, conforme necessário.</p> <p>Quando o cliente estiver pronto para fazer o checkout, ele clica no botão "finalizar compra". O sistema exibe o resumo do pedido, incluindo todos os itens no carrinho, o total da compra e opções de envio. O cliente fornece informações de entrega, incluindo endereço de envio e o método de entrega preferido.</p>

Fluxos alternativos

O sistema calcula o total final da compra, incluindo impostos e custos de envio. O cliente escolhe um método de pagamento e fornece as informações necessárias, como número do cartão de crédito e CVV.

O sistema valida as informações do cartão e processa a transação de pagamento. O cliente recebe uma confirmação da compra, exibindo um número de pedido único e os detalhes da transação. O sistema atualiza o estoque para refletir os itens comprados e envia uma confirmação de pedido para o cliente por e-mail.

Se o cliente não estiver autenticado, ele será redirecionado para a página de login antes de poder adicionar itens ao carrinho de compras. Se um produto estiver temporariamente esgotado, o sistema avisará o cliente e fornecerá a opção de ser notificado quando o produto estiver disponível novamente ou remover o item do carrinho. Se ocorrer um erro durante o processamento do pagamento, o sistema exibirá uma mensagem de erro e oferecerá opções para o cliente tentar novamente ou selecionar um método de pagamento diferente.

Pós-condições

Os produtos são reservados no estoque. O cliente recebe uma confirmação da compra por e-mail. O pedido é registrado no sistema para processamento e envio.

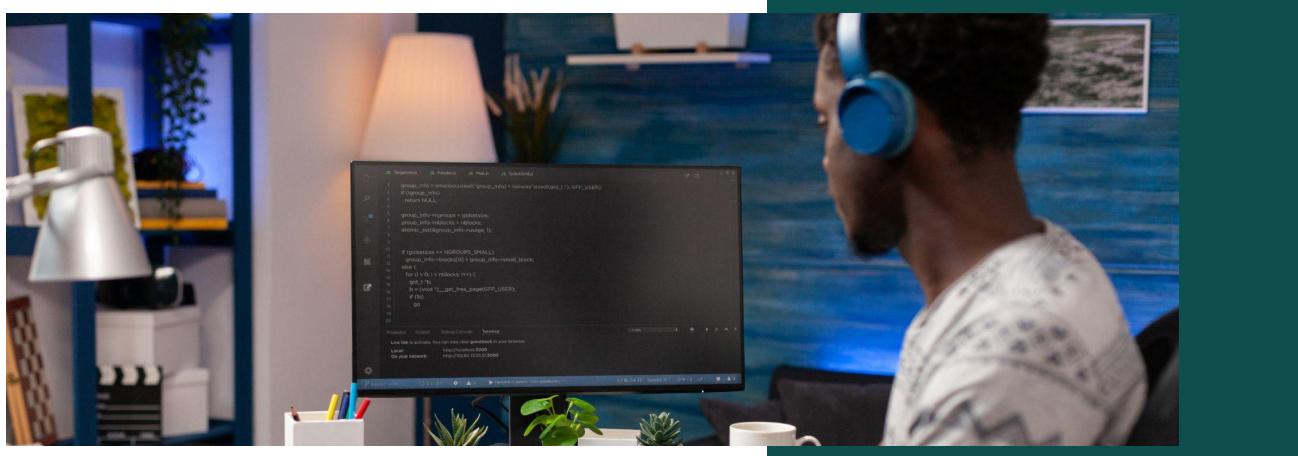
Regras de negócio

O sistema não permite que o cliente finalize uma compra se o carrinho de compras estiver vazio. O sistema aplica descontos ou promoções automaticamente, se disponíveis. Os produtos só podem ser adicionados ao carrinho de compras se estiverem disponíveis em estoque. As informações do cartão de crédito do cliente são criptografadas e armazenadas com segurança no sistema de pagamento.

Dados envolvidos	Informações do cliente: nome, endereço de entrega, endereço de e-mail. Informações do produto: nome, descrição, preço, quantidade em estoque. Informações do pedido: número de pedido, itens do pedido, subtotal, impostos, custos de envio.
Requisitos não funcionais	O sistema de <i>e-commerce</i> deve ser capaz de processar transações de compra de forma segura e eficiente. O tempo de resposta do sistema durante o processo de checkout deve ser rápido para garantir uma experiência do usuário satisfeita. A interface do usuário deve ser intuitiva e responsiva em diferentes dispositivos e navegadores.
Casos de teste associados	Teste para adicionar um item ao carrinho de compras. Teste para verificar se os produtos selecionados estão corretamente refletidos no carrinho de compras. Teste para verificar se as informações de pagamento são processadas corretamente.
Diagrama de sequência	Um diagrama de sequência pode ser criado para visualizar a interação entre o cliente, o sistema de <i>e-commerce</i> e o sistema de pagamento durante o processo de realização da compra.

Fonte: Elaborada pelos autores (2024).

Essa descrição detalhada fornece uma visão abrangente e precisa do caso de uso "realizar compra", incluindo todos os aspectos relevantes para a implementação e teste bem-sucedidos do sistema de *e-commerce*.



Até aqui, exploramos o diagrama de casos de uso em UML, uma ferramenta poderosa na modelagem de sistemas de software.

Começamos entendendo os conceitos fundamentais, como atores e casos de uso, e depois mergulhamos nos relacionamentos entre eles. Em seguida, passamos por um exemplo prático de um sistema de *e-commerce*, identificando atores, casos de uso e criando um diagrama para representar suas interações e destacamos a importância desses diagramas na engenharia de software.

Por fim, apresentamos a descrição de casos de uso, destacando que cada nível serve a um propósito específico ao longo do ciclo de vida do desenvolvimento de software.

A descrição informal é útil para comunicar conceitos-chave, enquanto a descrição típica fornece uma visão mais estruturada do caso de uso, e a descrição detalhada é necessária para orientar a implementação e o teste do sistema.

Ao escolher o nível de descrição apropriado, os desenvolvedores podem garantir que todos os aspectos do sistema sejam adequadamente documentados e compreendidos.

Acompanhe, adiante, como funciona a modelagem conceitual.

5. Modelagem conceitual

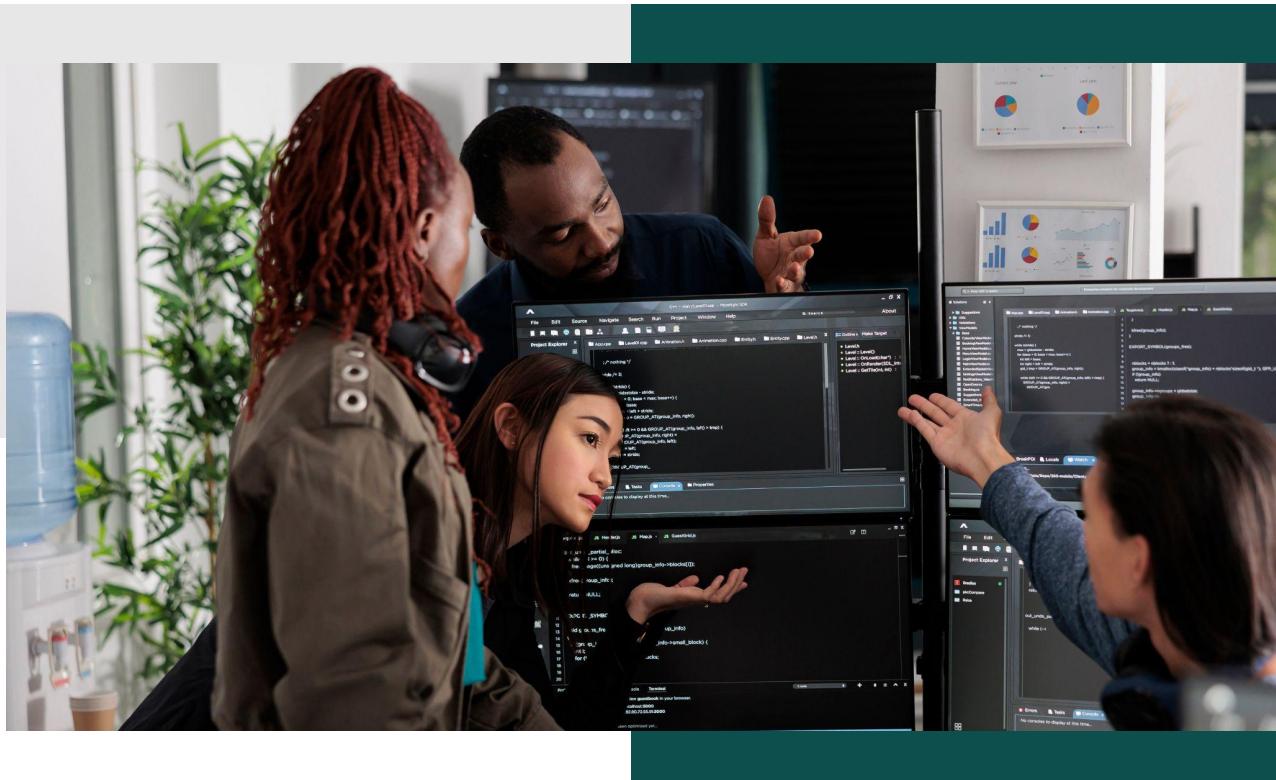
Neste tópico, serão abordados assuntos relacionados a conceitos, atributos, associações e organização do modelo.

A modelagem conceitual de sistemas é uma etapa fundamental no processo de desenvolvimento de software, em que se busca compreender e representar a estrutura e o funcionamento de um sistema de forma abstrata e de alto nível, isto é, visa à compreensão dos conceitos envolvidos no **domínio do problema** ou **domínio da aplicação**.

A propósito, o modelo conceitual tem por objetivo apresentar uma visão geral das entidades envolvidas no sistema, pois se trata da representação de conceitos no domínio do problema.

Essa abordagem permite à equipe de desenvolvimento e às partes interessadas entender a essência do sistema, sem se preocupar com os detalhes de implementação técnica.

O modelo conceitual se dá ainda durante a fase de análise, representando as classes no **domínio do problema** em questão, por exemplo, um sistema de controle de vendas ou um sistema de controle de lavação de veículos.



Nessa etapa, não se deve levar em consideração restrições inerentes à tecnologia a ser utilizada na solução de um problema. De acordo com Bezerra (2014), o modelo conceitual corresponde a uma representação abstrata do domínio da aplicação no qual se insere o sistema de software a ser desenvolvido. Segundo Wazlawick (2014), o modelo conceitual é uma representação da visão do usuário sobre a informação. Por outro lado, o modelo de classes de especificação, que deve ser desenvolvido durante a fase de projeto, é obtido por meio da adição de detalhes ao modelo conceitual conforme a solução de software escolhida, ou seja, o modelo conceitual praticamente antecede o modelo de classes de um projeto.

O principal objetivo da modelagem conceitual é a elaboração de uma representação clara e compreensível do sistema, servindo como um guia para o projeto e a implementação subsequentes. Para isso, são utilizadas técnicas e ferramentas específicas para identificar as entidades principais do sistema, os seus atributos, relacionamentos e comportamentos.

Um dos principais benefícios da modelagem conceitual é facilitar a comunicação entre os diferentes participantes envolvidos no projeto, incluindo clientes, usuários finais, analistas de sistemas e desenvolvedores.

Ao criar uma representação visual do sistema, é possível garantir que todas as partes tenham uma compreensão comum dos requisitos e das funcionalidades esperadas. Existem várias técnicas de modelagem conceitual disponíveis, cada uma com suas próprias características e aplicações. Entre as mais comuns, estão o diagrama de entidade-relacionamento (DER), e o modelo de classes, sendo este muito utilizado em sistemas orientados a objetos.

A modelagem conceitual é frequentemente realizada em conjunto com outras atividades de análise e projeto de sistemas, como a elicitação de requisitos, a definição de arquitetura e a validação de soluções propostas.

Em resumo, a modelagem conceitual de sistemas desempenha um papel crucial no desenvolvimento de software, fornecendo uma base sólida para a criação de sistemas complexos e de alta qualidade. Ao criar uma representação abstrata e comprehensível do sistema, os desenvolvedores podem garantir que as soluções propostas atendam às necessidades e expectativas dos usuários finais.



De acordo com Wazlawick (2014), na orientação a objetos, o modelo conceitual é representado pelo diagrama de classes da UML, no entanto, não se deve, nesta etapa, incluir quaisquer métodos. O modelo conceitual, essencialmente, deve mostrar conceitos, associações entre conceitos e atributos de conceitos. Resumidamente, no modelo conceitual, existem três tipos de elementos a serem considerados, sendo eles: conceitos, atributos e associações.

Conceito

Um conceito é uma ideia abstrata que representa uma classe de objetos ou entidades no sistema. Ele encapsula características comuns compartilhadas por esses objetos. No contexto do domínio do problema, o **conceito** nada mais é do que a representação de objetos envolvidos na aplicação, também chamados de entidades. Por exemplo, em um sistema de vendas, pode haver várias entidades ou instâncias de classes, tais como produtos, categorias de produtos, fornecedores, clientes e vendas, que por suas características serão representados pelas respectivas

classes de entidades **Categoria**, **Produto**, **Fornecedor**, **Cliente** e **Venda**. Tais representações são mostradas na figura a seguir e representam os conceitos do domínio do problema.

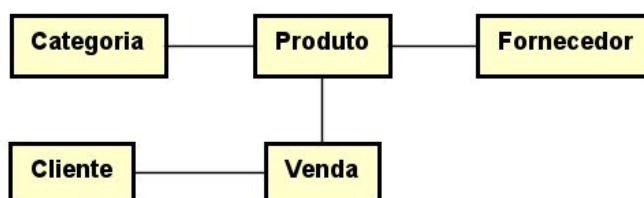
Conceitos preliminares do domínio do problema



Fonte: Elaborada pelos autores (2024).

A partir dos conceitos encontrados em um dado domínio de problema, é possível estabelecer um modelo conceitual preliminar (Wazlawick, 2014) que ainda não requer um alto nível de detalhamento. Isto é, apresenta-se a relação entre as entidades envolvidas sem a necessidade de identificá-las e tampouco de definir detalhes sobre atributos. Dessa forma, com base nas entidades supracitadas, é possível entender que produto tem uma categoria e que ele é fornecido por algum fornecedor. Por outro lado, quando ocorrer uma venda de produtos, um cliente estará envolvido na relação. Assim, a figura a seguir representa as relações entre os conceitos do sistema.

Associação preliminar das classes do modelo

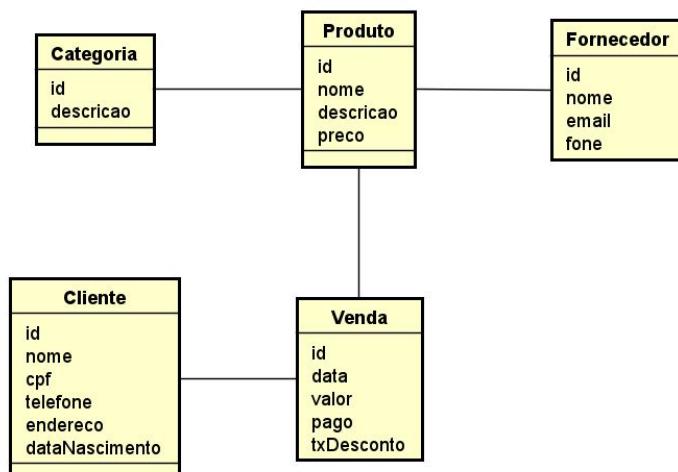


Fonte: Elaborada pelos autores (2024).

Atributos

As entidades envolvidas em um domínio de problema são caracterizadas pelos atributos que o identificam, sendo eles essenciais para o processo de classificação de objetos. Atributos são as características ou propriedades que descrevem os conceitos. Eles fornecem detalhes sobre as classes e são representados por variáveis dentro das classes. Por exemplo, a classe Produto pode ter atributos como **codigo**, **nome**, **descricao** e **preco**. A figura a seguir mostra um exemplo do modelo conceitual.

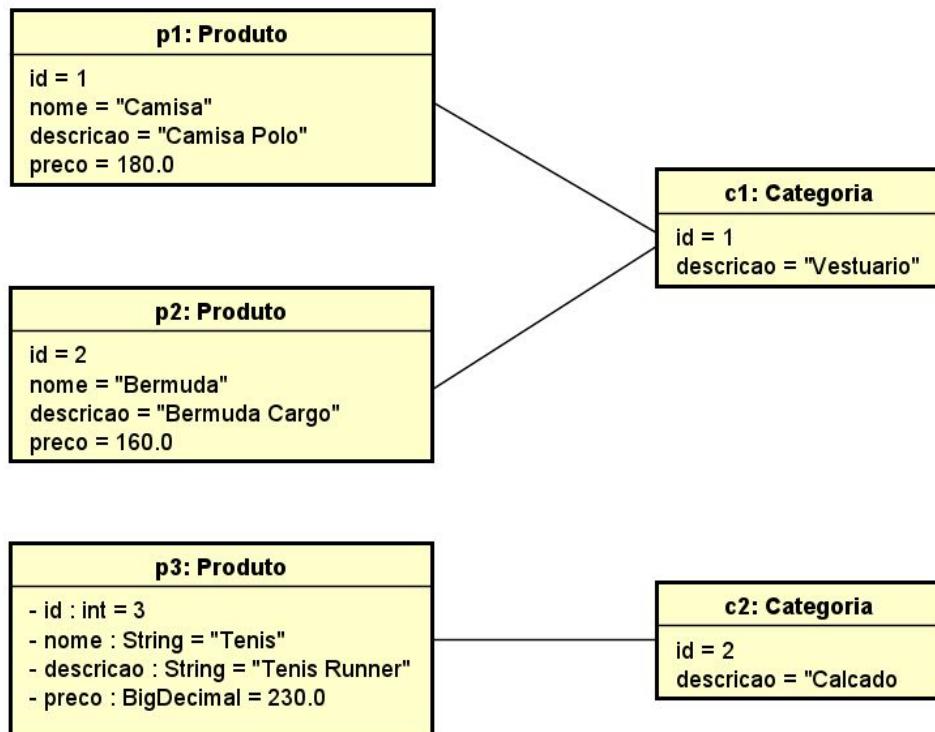
Atributos de classe definindo
as características do conceito



Fonte: Elaborada pelos autores (2024).

Para exemplificar, vale destacar que cada instância de classe terá os mesmos atributos especificados, porém, cada qual com seus valores representando o estado do objeto no sistema. Na figura a seguir, é possível observar um diagrama de objetos com três entidades produtos associadas a determinadas categorias.

Diagrama de objetos representando entidades do tipo produto e categoria



Fonte: Elaborada pelos autores (2024).

Resumidamente, os atributos de uma classe/um conceito oferecem detalhes específicos sobre a entidade, além de auxiliar na identificação entre diferentes instâncias dessa entidade.

Associação

Uma associação representa a relação entre duas ou mais entidades no sistema. Ela descreve como as entidades estão conectadas ou como interagem umas com as outras. De modo preliminar, é possível observar a associação de classes nas figuras intituladas “Associação preliminar das classes do modelo” e “Atributos de classe definindo as características do conceito”.

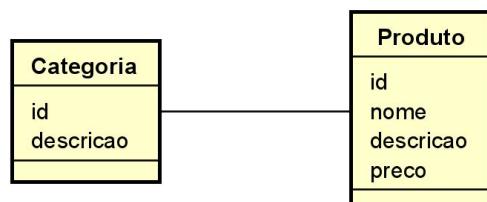
Em um sistema de vendas de produtos (*e-commerce*, por exemplo), a associação entre as entidades produto e categoria representa o fato de que um produto pode estar associado a uma categoria, e uma categoria, a muitos produtos.



Existem diversos tipos de associações que podem ser identificados durante a modelagem conceitual de sistemas. A seguir, são apresentados alguns dos tipos mais comuns de associação.

Associação simples: é a associação mais básica entre duas entidades. Ela indica que existe uma ligação direta entre os objetos das entidades envolvidas e é representada por uma linha contínua simples. Por exemplo, na modelagem do sistema de vendas, pode-se ter uma associação simples entre as entidades **produto** e **categoria**, indicando que um produto pertence a uma categoria. A figura a seguir apresenta um exemplo para este tipo de associação.

Associação simples entre produto e categoria



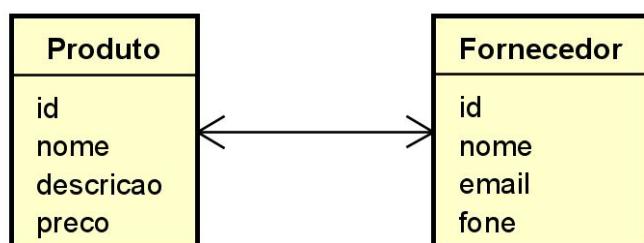
Fonte: Elaborada pelos autores (2024).

No tipo **associação bidirecional**, ambas as entidades estão conectadas uma à outra. Para exemplificar, no sistema de vendas, pode haver uma associação bidirecional entre as entidades **Produto** e **fornecedor**, indicando que um produto tem relação e conhece seu fornecedor, assim como um fornecedor pode conhecer e fornecer uma gama de produtos. Um exemplo para esse tipo de associação está na figura a seguir. Note que, na figura, há setas em cada extremidade da linha indicando o sentido da relação, remetendo a uma ligação bidirecional.

O uso das setas também indica a naveabilidade da leitura e a compreensão da relação entre as classes, isto é, há a necessidade de se compreender a relação nos dois sentidos, em que cada produto *conhece* o seu fornecedor e cada fornecedor *conhece* os produtos que fornece.

Em termos de implementação de código, isso implicará a declaração de variáveis em ambas as classes.

Associação bidirecional entre produto e fornecedor



Fonte: Elaborada pelos autores (2024).

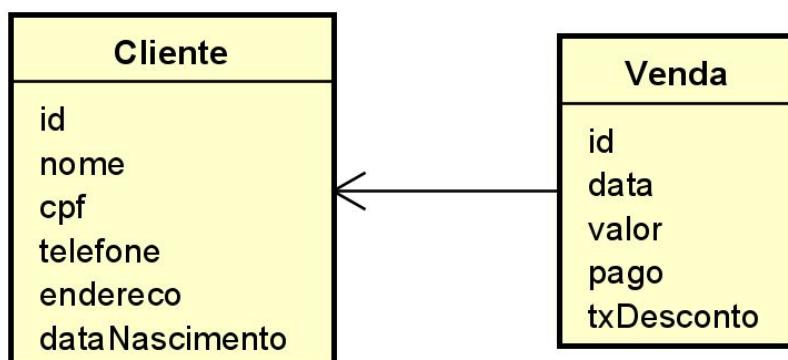
A associação bidirecional pode ser assumida também quando houver, no diagrama, uma linha contínua sem qualquer seta indicando navegabilidade, tal qual mostrada na figura M5, isto é, representada por uma **associação simples**.

Já na **associação unidirecional**, a vinculação entre as entidades é unilateral, ou seja, apenas uma das entidades está conectada à outra.

Na associação unidirecional, uma seta de navegabilidade indica o sentido da relação, apontando a interpretação da associação.

Por exemplo, em um sistema de pedidos on-line, pode haver uma associação unidirecional entre as entidades Venda e Cliente, indicando que uma Venda está associada a um Cliente, mas o Cliente pode não estar diretamente associado ao Pedido. Considerando o exemplo, ainda, assume-se que a **venda conhece o cliente**, mas o contrário não requer interpretação. A figura a seguir esboça esse tipo de associação.

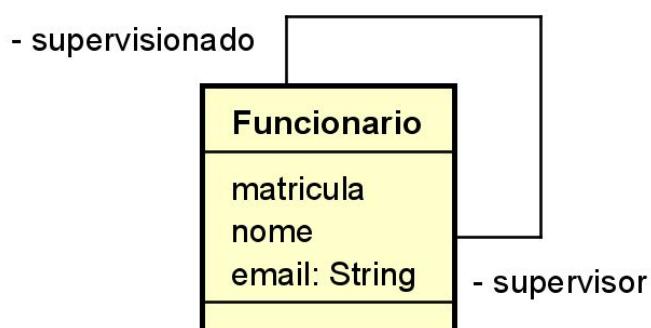
Associação unidirecional entre venda e cliente, caracterizada pela seta de navegabilidade de Venda para Cliente



Fonte: Elaborada pelos autores (2024).

A **associação reflexiva**, por sua vez, ocorre quando uma entidade está associada a ela mesma. Por exemplo, em um sistema de gerenciamento de funcionários, pode haver uma associação reflexiva na entidade funcionário para representar o relacionamento de supervisão, em que um funcionário pode supervisionar outros funcionários.

Associação reflexiva de funcionário com os papéis supervisionado e supervisor



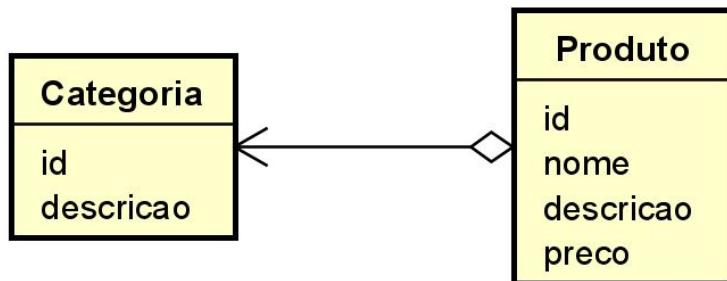
Fonte: Elaborada pelos autores (2024).

Na **associação de agregação**, uma entidade é composta de outras entidades menores. Esse tipo de associação representa a relação **tem-um** ou **parte-de**, em que a classe que agrupa pode conter um ou mais entidades de outra classe, com a característica de que cada entidade envolvida pode existir de forma independente.

Essa relação é representada por um losango vazio na linha de associação junto à classe (conceito) agregadora.

Por exemplo, em um sistema de vendas, uma classe **produto** pode agregar uma classe **categoria**, em que uma entidade **categoria** pode existir separadamente do **produto**. A figura a seguir mostra essa representação.

Associação agregação de Produto com Categoria



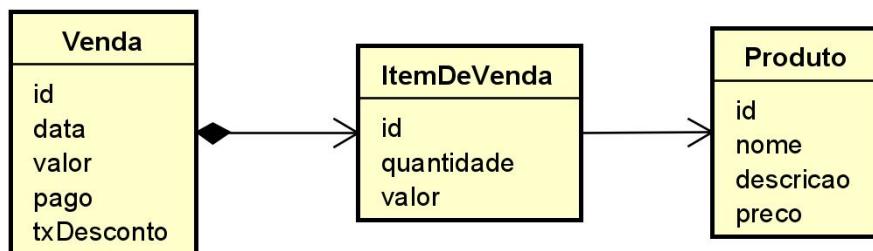
Fonte: Elaborada pelos autores (2024).

A **associação de composição** é similar à associação de agregação, exceto que ela se demonstra mais forte, isto é, mantendo uma relação de dependência de vida. Essa associação também é conhecida como **todo-parte**.

Na associação de composição, uma entidade é constituída por outras entidades menores. No entanto, a diferença principal é que na associação de composição, a existência das entidades menores depende da entidade composta. Neste sentido, se uma entidade que é composta por outras for destruída, todas as entidades menores que a compõem serão destruídas também.

Graficamente, essa relação é representada por um losango fechado na linha de associação junto à classe (conceito) que leva as entidades associadas. Por exemplo, em um sistema de vendas, uma entidade **venda** é composta pelos itens de venda, assim sendo, se **venda** for destruída, todos os itens de venda também serão. A figura a seguir mostra esse tipo de associação.

Associação de composição de Venda com ItemDeVenda



Fonte: Elaborada pelos autores (2024).

Há, ainda, a multiplicidade das associações, um aspecto que você vai conhecer a seguir

Multiplicidade das associações

As associações são caracterizadas pela multiplicidade da relação entre entidades. Ela define o número de instâncias de uma classe que podem estar relacionadas com uma única instância de outra classe. Ela é representada por números ou intervalos mínimos e máximos de objetos que podem participar de uma relação.



Veja, a seguir, as formas de especificação da multiplicidade:

1 -

Indica uma relação um para um. Por exemplo, um **produto** tem uma **categoria**, ou seja, está associado a uma **categoria**.

0..1

Significa zero ou um. Neste caso, uma instância de uma classe A pode estar associada a, no máximo, uma da classe B.

1..*

Relação um para muitos. Neste caso, uma instância pode estar associada a uma ou mais instâncias de outro tipo associado a ele. Por exemplo, uma venda deve ter um ou mais itens de venda.

0..*

Essa multiplicidade indica zero ou muitos. Um fornecedor pode estar cadastrado, mas não necessariamente fornecer produtos, logo, é possível afirmar que um fornecedor pode fornecer zero ou mais produtos.

Só o * (asterisco) indica uma representação de muitos, mas é usado quando não há necessidade de especificar o limite inferior. Por exemplo, um produto pode estar em muitas vendas.

A figura intitulada “Modelo conceitual de um sistema de controle de vendas contemplando os tipos de associações, multiplicidade e relação por herança” apresenta o diagrama completo com as multiplicidades descritas nesta seção.

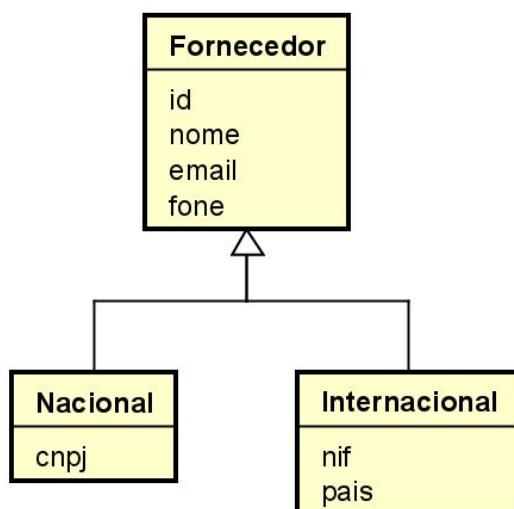
Herança

É um tipo de relação entre classes e não se caracteriza como uma associação. Na herança, uma classe herda os conceitos de outra classe. A classe herdeira é chamada de subclasse (classe especializada), enquanto a classe herdada se chama superclasse (classe genérica).

Em outras palavras, no modelo conceitual, esse processo de abstração se chama de generalização e especialização.

Por exemplo, no sistema de vendas, podemos assumir dois subtipos de **fornecedor**, o **nacional** e o **internacional**. Neste sentido, diz-se que **nacional** e **internacional** são entidades que herdam características comuns da superclasse **fornecedor**. Graficamente, a herança é representada por uma seta fechada na direção da superclasse. A figura a seguir apresenta esse tipo de relação.

Relação de classes por herança com a superclasse fornecedor e as subclasses nacional e internacional.



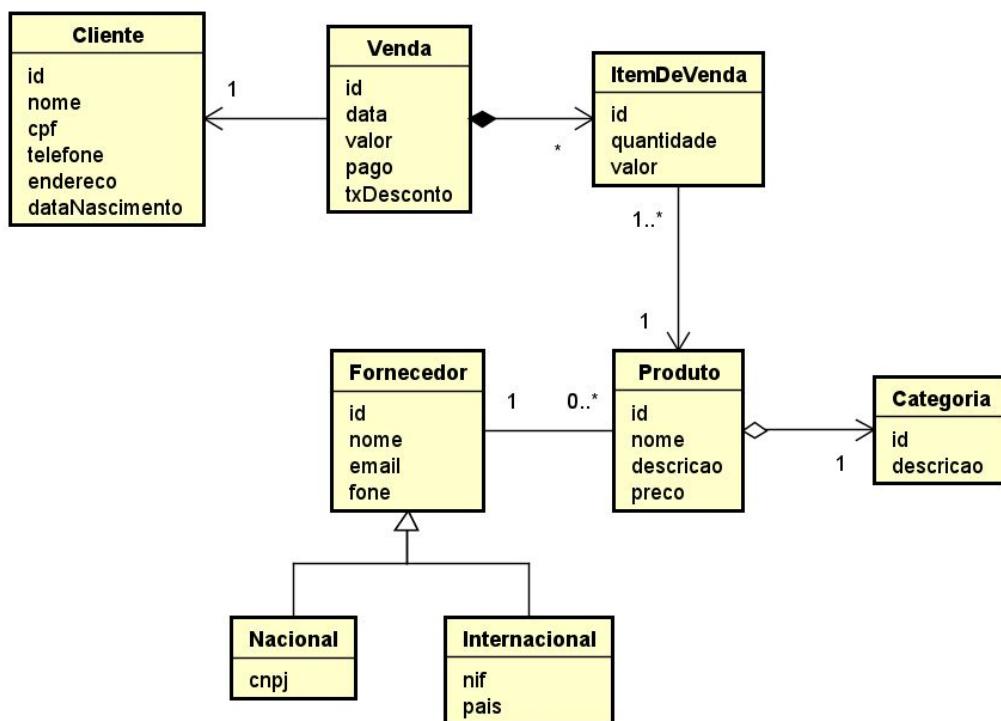
Fonte: Elaborada pelos autores (2024).

Confira, a seguir, o último diagrama.

Diagrama do modelo conceitual das classes exemplificadas

Observe, agora, o diagrama do modelo conceitual das classes envolvidas em um sistema de vendas contendo atributos, relações por associações, herança e multiplicidade.

Modelo conceitual de um sistema de controle de vendas contemplando os tipos de associação, multiplicidade e relação por herança



Fonte: Elaborada pelos autores (2024).

Portanto, essa figura apresenta todas as classes e todos os relacionamentos vistos nesta Unidade Curricular.

Finalizando

Nesta Unidade Curricular, você viu o que são requisitos, seus tipos e as principais técnicas para elucidar os requisitos. Viu, também, a UML, linguagem de modelagem universal que facilita a documentação do software em todas as suas fases.

Conhecemos a ferramenta de montagem Astah e começamos entendendo os conceitos fundamentais dos diagramas de casos de uso, como atores e casos de uso, e depois mergulhamos nos relacionamentos entre eles. Em seguida, passamos pelo exemplo prático de um sistema de *e-commerce*, identificando atores, casos de uso, e criando um diagrama para representar suas interações, e destacamos a importância desses diagramas na engenharia de software.

Além disso, detalhamos a descrição de casos de uso, em que se enfatizou que cada nível serve a um propósito específico ao longo do ciclo de vida do desenvolvimento de software. A descrição informal é útil para comunicar conceitos-chave, enquanto a descrição típica fornece uma visão mais estruturada do caso de uso, e a descrição detalhada é necessária para orientar a implementação e teste do sistema. Ao escolher o nível de descrição apropriado, os desenvolvedores podem garantir que todos os aspectos do sistema sejam adequadamente documentados e compreendidos. Por fim, vimos o modelo conceitual em que se busca compreender e representar a estrutura e o funcionamento de um sistema de forma abstrata e de alto nível, isto é, visa à compreensão dos conceitos envolvidos no domínio do problema ou no domínio da aplicação.

Esses conceitos serão aplicados e complementados nas próximas Unidades Curriculares, visando ao desenvolvimento completo do sistema para controle de lavação de veículos.

Referências

ASTAH. Products. **Astah is a modeling tool.** [2024]. Disponível em: <https://astah.net/products/>. Acesso em: 20 maio 2024.

BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML.** 3. ed. Rio de Janeiro: GEN LTC, 2014.

BOOCH, G., RUMBAUGH, J.; JACOBSON, I. **UML:** Guia do Usuário. 2. ed. Campus, 2006.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **Unified Modeling Language User Guide.** 2. ed. Boston: Addison-Wesley, 2005. The Addison-Wesley Object Technology Series.

DENNIS, A.; WIXOM, B. H.; ROTH, R. M. **Análise e Projeto de Sistemas.** São Paulo: Grupo GEN, 2014. E-book. ISBN 978-85-216-2634-3. Disponível em: <https://app.minhabiblioteca.com.br/#/books/978-85-216-2634-3/>. Acesso em: 20 maio 2024.

GUEDES, G. T. A. **UML 2:** Uma Abordagem Prática. 3. ed. São Paulo: Novatec, 2018.

JACOBSON, I.; MAGNUS, C.; PATRICK, J.; GUNNAR, O. **Object-Oriented Software Engineering.** Boston: Addison-Wesley, 1992.

LARMAN, C. **Applying UML and Patterns:** An Introduction to Object-Oriented Analysis and Design and Iterative Development. Nova Jersey: Prentice Hall, 2004.

LEDUR, C. L. **Análise e projeto de sistemas.** Porto Alegre: Grupo A, 2018. E-book. ISBN 9788595021792. Disponível em: <https://app.minhabiblioteca.com.br/#/books/9788595021792/>. Acesso em: 20 maio 2024.

OBJECT MANAGEMENT GROUP. **OMG Modeling Language.** Superstructure. Versão 2.2. 2009. Disponível em: <https://www.omg.org/spec/UML/2.2/Superstructure/PDF>. Acesso em: 20 maio 2024.

OBJECT MANAGEMENT GROUP. **UML.** Unified Modeling Language. Versão 2.5.1. 2017. Disponível em: <https://www.omg.org/spec/UML/>. Acesso em: 20 maio 2024.

SCHACH, S. **Engenharia de Software:** os paradigmas clássico e orientado a objetos. 7. ed. Porto Alegre: McGraw-Hill, 2009.

SERAO, L. A. J. **Desafios na concepção de projetos de TI:** Falta de coesão na comunicação. 2020. LinkedIn. Disponível em: <https://www.linkedin.com/pulse/desafios-na-concep%C3%A7%C3%A3o-de-projetos-ti-falta-coes%C3%A3o-juc%C3%A1-serao/>. Acesso em: 20 maio 2024.

STEINPICHLER, D.; KARGL, H. **Project management with UML and EA - Enterprise Architect.** 8. ed. Victoria: Sparx Systems, 2011. Disponível em: https://edisciplinas.usp.br/pluginfile.php/1787945/mod_resource/content/1/UML_Basics.pdf. Acesso em: 20 maio 2024.

WAZLAWICK, R. S. **Análise e Design Orientados a Objetos para Sistemas de Informação:** Modelagem com UML, OCL e IFML. São Paulo: Grupo GEN, 2014. E-book. ISBN 9788595153653. Disponível em: <https://app.minhabiblioteca.com.br/#/books/9788595153653/>. Acesso em: 20 maio 2024.

