

Resumo para Prova 1:

Materias que vão cair na prova: Desde a Aula 1 até a Aula 8, utilizamos a ferramenta Astah durante as aulas.

Material 1: Conceitos Iniciais

Conceitos de Orientação Objetos

Observando duas fotos de carros diferentes, um antigo e outro mais novo.

Ainda assim, são carros. E sendo carros, há questões que podem ser reaproveitadas entre eles - tais como Portas, Quantidade de Pneus, Volante, etc.

São atributos "comuns" de um carro, ou seja, o carro pode ser diferente mas esses atributos estarão presentes.

No entanto, mesmo entre as questões genéricas, há os detalhes, as especificidades de cada carro, como potência do motor (embora ambos tenham motor), valor do seguro, entre outros, então mesmo que existam características em comum, ainda existem valores

Por isso, podemos dizer que Carro é uma Classe, mas um Ford Ka, um Toyota Corolla, uma VW Kombi são Instâncias desta Classe - ou seja, um Objeto, ou seja - a generalização é caracterizada pelo objeto, enquanto as especificidades são como instâncias do mesmo.

Porque a Orientação a Objeto existe ?

Ela existe para tornar o código mais fácil de manter, o que significa ser fácil para mater:

- Possibilidade de reutilizar o código sem copia e cola

- Entendimento de quem for ler o código além do autor original
- Alteração deve ser fácil de ser realizada

Objetos:

Definição de Objetos:

Um objeto é qualquer coisa existente no mundo real, em formato concreto ou abstrato, ou seja que exista fisicamente ou apenas conceitualmente.(EXEMPLO: Aluno, professor, mesa, cadeira, caneta, automóvel, estoque, botão...)

Um objeto é uma instância de uma classe. Ou seja, é algo concreto criado a partir do molde que a classe define.

Objetos possuem ATRIBUTOS

Atributos:

Os atributos identificam o estado de um objeto, são características ou estado de um objeto.()EXEMPLO:

OBJETO: Cliente | Atributo: Nome, endereço, sexo, data de nascimento)

São valores específicos e únicos daquele objeto.

Além dos atributos os objetos possuem comportamentos, denominados como OPERAÇÕES

Operações

Ações que o objeto pode executar.

Uma operação é um conceito mais abstrato que representa uma ação que pode ser realizada por um objeto. É definida na interface ou na especificação de uma classe, indicando o que o objeto pode fazer, sem detalhar como isso será

implementado.

Um **método** é a implementação concreta de uma operação. É o código real que executa a funcionalidade quando invocado. Os métodos definem o comportamento específico de um objeto.

Métodos

Os métodos de uma classe manipulam somente as estruturas de dados daquela classe, ou seja, não podem acessar diretamente os dados de outra classe.

Os métodos representam ações que os objetos podem realizar. Eles funcionam como funções dentro da classe e podem modificar atributos ou executar ações .

Um Método é a implementação de uma operação, ou seja, sua representação em código

Mensagens

Uma Classe tem conhecimento dos dados de outra, pela solicitação de serviços (execução de operações) denominadas **mensagens**

Classe:

Uma classe é um modelo ou um molde para criar objetos. Ela define quais características (atributos) e comportamentos (métodos) dos objetos terão.

A classe é a caixa que guarda as informações do objeto Quando identificamos características e operações similares em objetos distintos, estamos realizando sua classificação, ou seja, identificando classes.

Objeto 1 : Lápis de escrever

Objeto 2 : Lápis de escrever

Objeto 3 : Lápis de escrever

Classe: Lápis (Obj 1, Obj2, Obj3)

Uma classe é uma representação de um conjunto de objetos que compartilham a mesma estrutura de atributos, operações e relacionamentos, dentro de um mesmo contexto (semântica).



é dos
lips

Lápis
classe

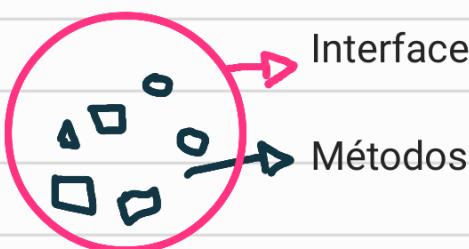
Encapsulamento

- O encapsulamento é agrupar os dados e métodos por suas responsabilidades.
- Prática de esconder como um objeto executa suas operações;
- Trás muitos benefícios pois o cliente não precisa saber como um objeto funciona internamente;

Interface

A interface serve como intermediaria entre a classe e o mundo externo, protegendo os usuários dessa classe de qualquer alterações futura. As alterações na classe são

feitas de modo transparente para o usuário.
Uma interface é o conjunto de métodos que você pode executar em um determinado objeto



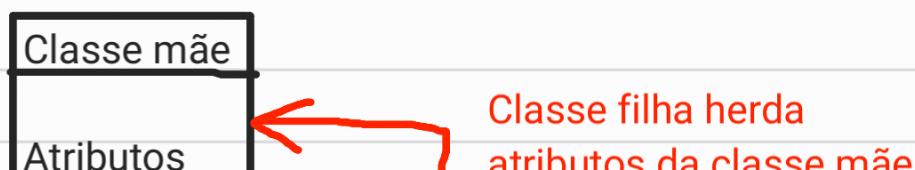
Generalização e especialização

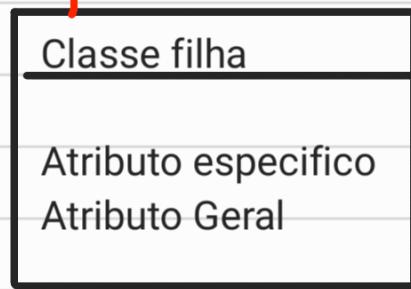
Ao separarmos da classe funcionários os atributos mais genéricos criando a classe pessoa, estamos realizando uma generalização. Ao contrário, quando estamos diante da classe pessoa e separarmos alguns atributos mais específicos, para serem colocados na classe funcionário estamos realizando uma especialização.

Herança

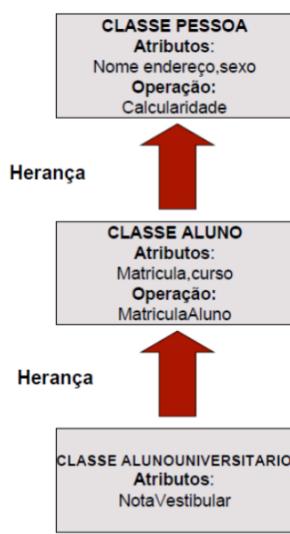
Se os objetos tem atributos em comum, mesmo sendo de classes diferentes, como por exemplo, uma empresa tem classe funcionário, classe cliente, os dois são pessoas automaticamente possuem algumas características semelhantes, podemos agrupar essas características.

Podemos criar uma classe genérica/classe mãe/ou superclasse, essa classe vai conter os atributos que servem para qualquer situação e depois podemos incluir classes específicas com atributos específicos, que são denominadas como classe filha/subclasse.

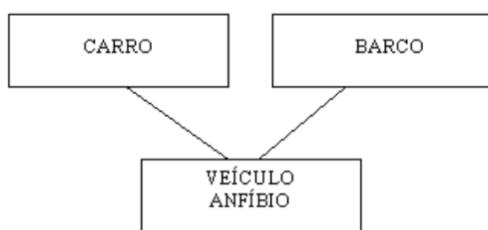
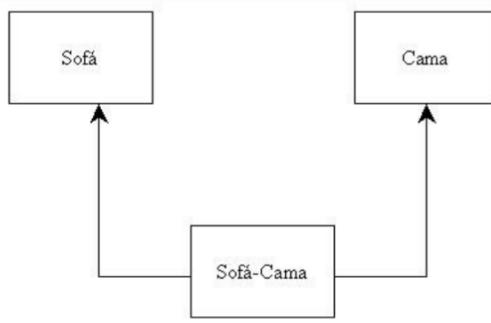




Exemplo de Herança:



Existe um tipo especial de herança que permite que uma classe possua mais de uma superclasse e herde características de todos os seus ancestrais. Trata-se da herança múltipla. Uma característica proveniente da mesma classe ancestral encontrada em mais de um caminho é herdada apenas uma vez. Uma classe com mais de uma superclasse é denominada classe de junção.

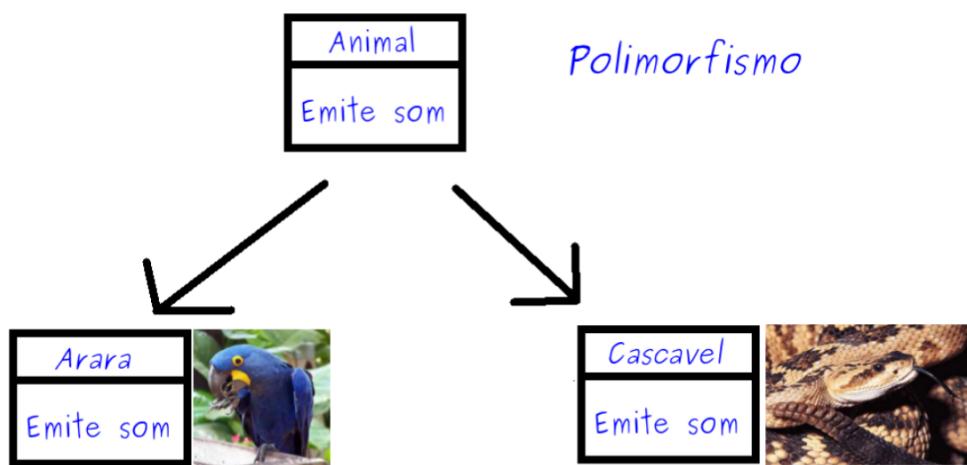


#POLIMORFISMO

No POO, o polimorfismo permite que os métodos tenham comportamentos diferentes dependendo do objeto. Como a programação estruturada não tem aulas, podemos simular isso usando ponteiros para funções.

Polimorfismo – Permite que métodos com o mesmo nome tenham comportamentos diferentes dependendo do contexto.

Pense numa operação que seja implementada nas classes filhas de forma diferente que na classe mãe. Isso é possível com o polimorfismo. Uma operação pode ter implementações diferentes em diversos pontos da hierarquia de classes, desde que mantenham a mesma assinatura (quantidade e tipos de argumentos e tipo do valor resultante). As operações das subclasses prevalecerão sobre as da superclasse



Material 2: Diagrama de Classes

UML- Unified Modeling Language

Conceitos:

A UML consiste de um certo número de elementos gráficos

que se combinam para formar diagramas. Como a UML é uma linguagem, ela possui regras para combinar estes elementos nos diversos diagramas.

O objetivo dos diagramas é apresentar múltiplas visões do sistema, sendo essas múltiplas visões chamadas de modelo.

Um modelo UML descreve o que o sistema fará mas não diz nada sobre implementar o sistema.

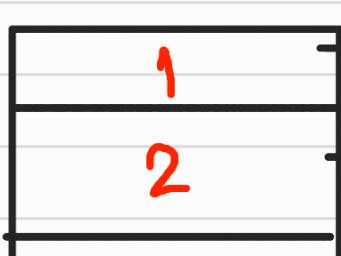
Uma **classe** é uma categoria ou grupo de (coisas) que possuem o mesmo atributo e comportamento.

Nesta linha de raciocínio a classe máquina de lavar roupa conterá elementos que possuem **atributos** como nome, modelo, número de série e capacidade e deverá apresentar comportamento que inclui as **operações** como: "aceitar roupas", "aceitar sabão", "ligar", "desligar".

Diagrama de Classes: Classe, Representação e Definições

Representação

- 1- A parte superior contém o nome da classe
- 2- A parte do meio contém os atributos da classe
- 3- A parte de baixo contém as operações(métodos) da classe



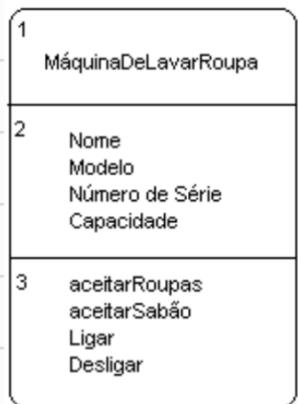
- 1- A parte superior contém o nome da classe
- 2- A parte do meio contém os atributos da classe
- 3- A parte de baixo contém as operações(métodos) da classe

3



3- A parte de baixo contém as operações(métodos) da classe

Exemplo:



1 - NOME

- Nomes curtos com letras e dígitos (ex.: ``Lugar'', ``DataReserva'').
- Tudo junto
- sem acentuação

2 - Propriedades /atributos

- informações específicas relacionadas àquela classe
- Cor, altura, tamanho, largura, modelo

3 - Métodos

- São ações que os objetos de uma classe podem realizar.
- Ex: Latir , correr , sentar , comer, etc.

Podemos dizer que os diagramas de classes são os principais diagramas estruturais da UML pois ilustram as classes , interfaces e relacionamentos entre elas.

REPRESENTAR VISIBILIDADE DOS ATRIBUTOS (encapsulamento)

- + **público** - visível em qualquer classe
- # **protegido** - qualquer descendente pode usar
- **privado** - visível somente dentro da classe

RELACIONAMENTO ENTRE CLASSES:

Os diagramas de classes ilustram atributos e operações de uma classe e as restrições como que os objetos podem ser conectados ; descrevem também os tipos de objetos no sistema e os relacionamentos entre estes objetos que podem ser : **associações e abstrações**.

Os objetos têm relações entre eles: um professor ministra uma disciplina para alunos numa sala, um cliente faz uma reserva de alguns lugares para uma data,

Essas relações são representadas também no diagrama de classe.

UML reconhece três tipos mais importantes de relações:

- **Dependência**
- **Associação**
- **Generalização (ou herança)**.

ASSOCIAÇÃO

São relacionamentos estruturais entre instâncias e especificam que objetos de uma classe estão ligados a objetos de outras classes. Podemos ter associação unária , binária , etc.

- **AGREGAÇÃO (Permanece vivo mesmo sem o todo)**

tipo de associação (é parte de, todo/parte) onde o objeto parte é um atributo do todo ; onde os objetos partes somente são criados se o todo ao qual estão agregados seja criado.

- **COMPOSIÇÃO(Morre se o todo morrer)**

Relacionamento entre um elemento (o todo) e outros elementos (as partes) onde as partes só podem pertencer ao todo e são criadas e destruídas com ele.

DEPENDENCIA

São relacionamentos de utilização no qual uma mudança na especificação de um elemento pode alterar a especificação do elemento dependente. A dependência entre classes indica que os objetos de uma classe usam serviços dos objetos de outra classe.

GENERALIZAÇÃO/HERANÇA

Relacionamento entre um elemento mais geral e um mais específico. Onde o elemento mais específico herda as propriedades e métodos do elemento mais geral.

O diagrama de classes depende de um levantamento de requisitos prévios, para que o diagrama possa ser construído

- Definição dos objetos do sistema : Paciente , agenda , dentista , serviço , contrato , consulta , pagamento , etc..

- Definição dos atores do sistema : paciente, dentista ,secretária
- Definição e detalhamento dos casos de uso: marca consulta , confirmar consulta , cadastrar paciente , cadastrar serviços , etc.
- Definição das classes : paciente , dentista , exame , agenda ,serviço

Material 3: Diagrama de classe no astah

Material 4: Generalização, agregação, composição e dependência.

Generalização:

Definição

- É a capacidade de se criar superclasses que encapsulam estrutura e/ou comportamento comuns a várias subclasses.

Procedimentos

- Os procedimentos para se obter a generalização são:
- Identificar similaridades de estrutura/comportamento entre várias Classes.
- Criar a superclasse para encapsular a estrutura comportamento comum.
- As classes originais passam a ser subclasses da nova superclasse criada.

Herança

É uma hierarquia de abstrações na qual uma subclasse

herda a estrutura e/ou comportamento de uma ou mais superclasses.

Tipos de Herança

- Herança simples é quando uma subclasse herda estrutura e/ou comportamento de uma única superclasse.
- Herança múltipla é quando uma subclasse herda estrutura e/ou comportamento de mais de uma superclasse.

IMPORTANTE: Herança é uma relação entre Classes de Objetos, e não uma relação entre instância das Classes.

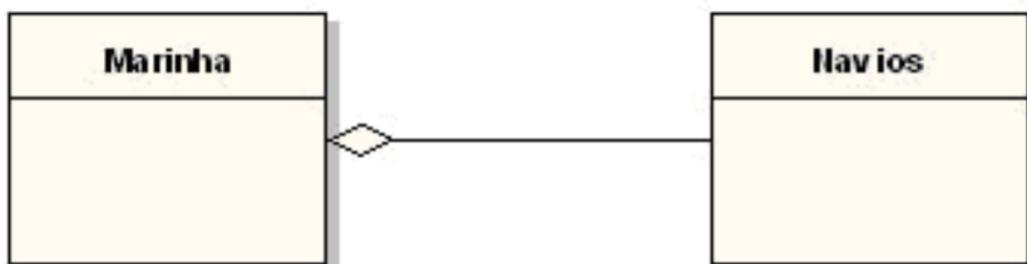
Agregação



Definição: É uma forma especializada de associação na qual um todo é relacionado com suas partes. Também conhecida como relação de conteúdo.

Como representamos uma Agregação?

É representada como uma linha de associação com um diamante junto à Classe agregadora. A multiplicidade é representada da mesma maneira que nas associações.



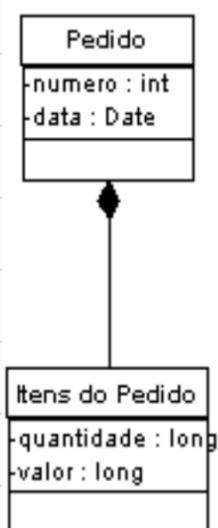
Composição



Definição: É uma agregação onde uma classe que está

contida na outra "vive" e constitui a outra. Se o objeto da classe que contém for destruído, as classes da agregação de composição serão destruídas juntamente, já que as mesmas fazem parte da outra.

Como representamos uma Agregação Composição?
É representada como uma linha de associação com um diamante preenchido junto à Classe agregadora.
A multiplicidade é representada da mesma maneira que nas associações.



Dependência →

Definição: Uma dependência indica a ocorrência de um relacionamento semântico entre dois ou mais elementos do modelo, onde uma classe cliente é dependente de alguns serviços da classe fornecedora, mas não tem uma dependência estrutural interna com esse fornecedor [Furlan, 1998]

Como funciona?

Por exemplo: Uma mudança no elemento independente irá afetar o modelo dependente. Como no caso anterior com generalizações, os modelos de elementos podem ser

uma classe, um pacote, um use-case e assim por diante.

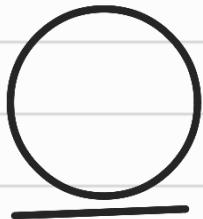
Material 5: Diagrama de classes : Esterótipos

Esterótipo: É um mecanismo para a extensão da UML.

Facilita a vida do desenvolvedor:

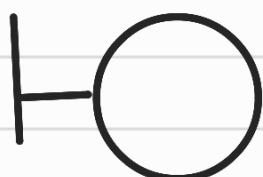
Entity

- Normalmente possuem vida longa(persistência).
- O comportamento não depende da forma do ambiente (dos diversos tipos possíveis de interface, por exemplo).
- Responsáveis pelas informações do negócio.
- Podem ser utilizadas em mais de uma aplicação.



Boundary

- Responsáveis pela comunicação do sistema com os atores
- interface com usuário
- Comunicação com outros sistemas



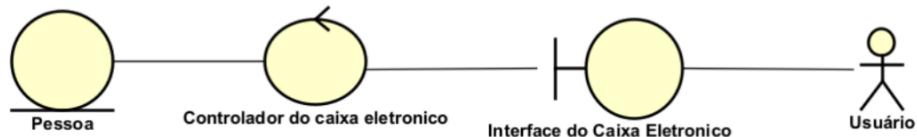
Control

Classes de Controle modelam o comportamento sequencial de um caso de uso. Classes de Controle coordenam os eventos necessários para realizar o comportamento especificado no

Caso de Uso. Você pode pensar em uma Classe de Controle como a responsável por “rodar” o Caso de Uso – elas representam a dinâmica do Caso de Uso. Classes de Controle são, tipicamente, dependentes da aplicação.

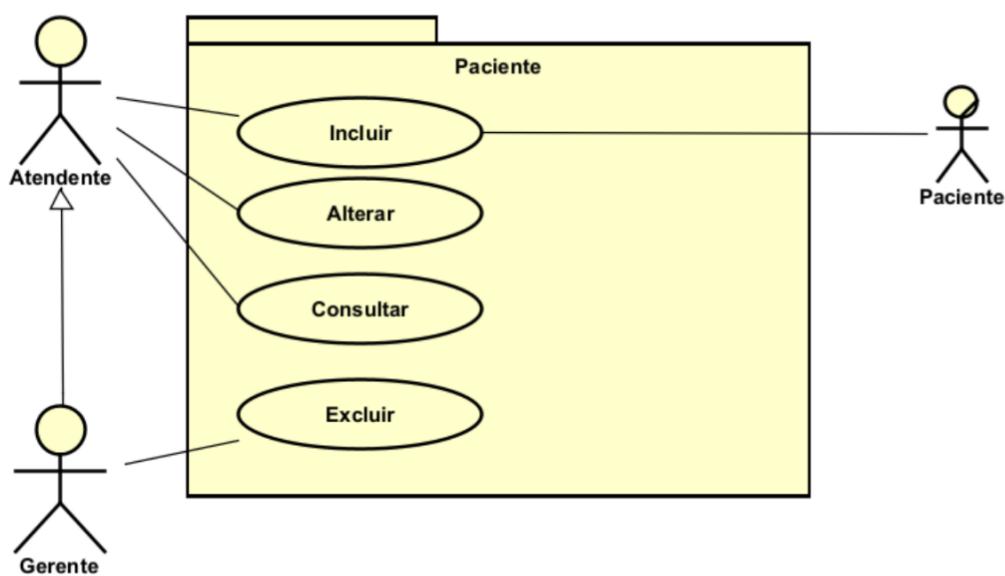


Exemplo:

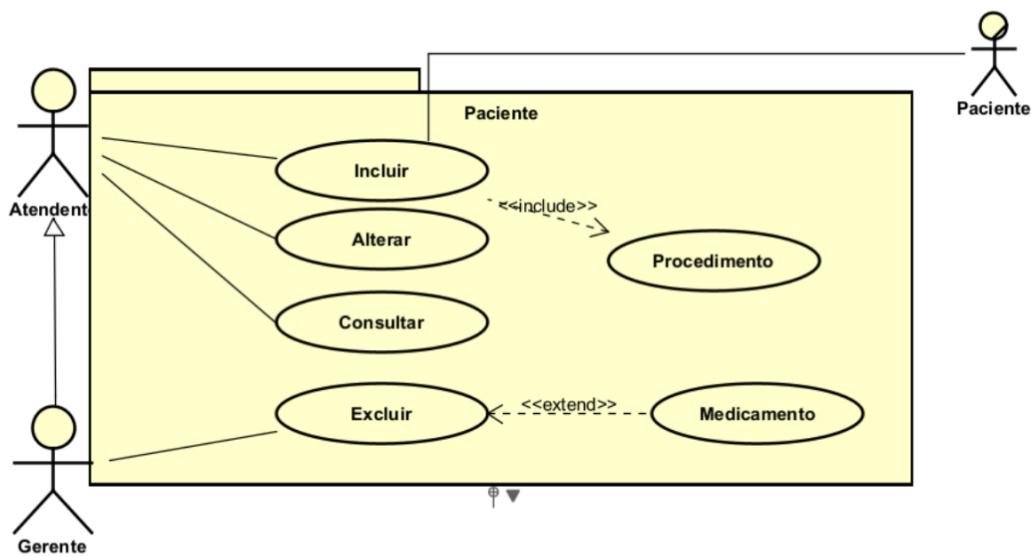


Boundary nunca liga direto com entity, obrigatoriamente tem que passar pelo control.

Bunisses actor: Não coloca a mão, ele só está participando da ação, ele só existe.



Include/Extende



Include: Obrigatório

Extend: Opicional

Material 6: Diagrama Sequencial

Diagrama sequencial:

Este diagrama procura determinar a sequência de eventos que ocorrem em um determinado processo, ou seja, quais condições devem ser satisfeitas e quais métodos devem ser disparados entre os objetos envolvidos e em que ordem durante um processo específico.

O diagrama de sequência baseia-se no Diagrama de caso de uso. No entanto deve-se ter em mente que o fato de haver um único Diagrama de casos de uso não significa em absoluto que deva haver um único diagrama de seqüência.

- **Atores**
- **Objetos**

- Linha de Vida
- Foco de Controle ou Ativação
- Mensagens ou Estímulos
- Mensagens de Retorno
- Auto Chamadas
- Condições

Atores

São exatamente os mesmos descritos no Diagrama de Casos de Uso, ou seja, entidades externas que interagem com o sistema e que solicitam serviços gerando dessa forma eventos que iniciam processos



Cliente

Autor normal: quem faz contato com a interface

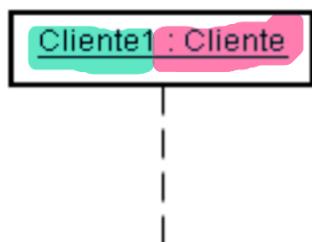


: Paciente

Não coloca a mão, ele só está participando da ação, ele só existe.

Objetos

Representam as instâncias das classes envolvidas no processo.



Nome do objeto

Seguido por ":"

Nome classe

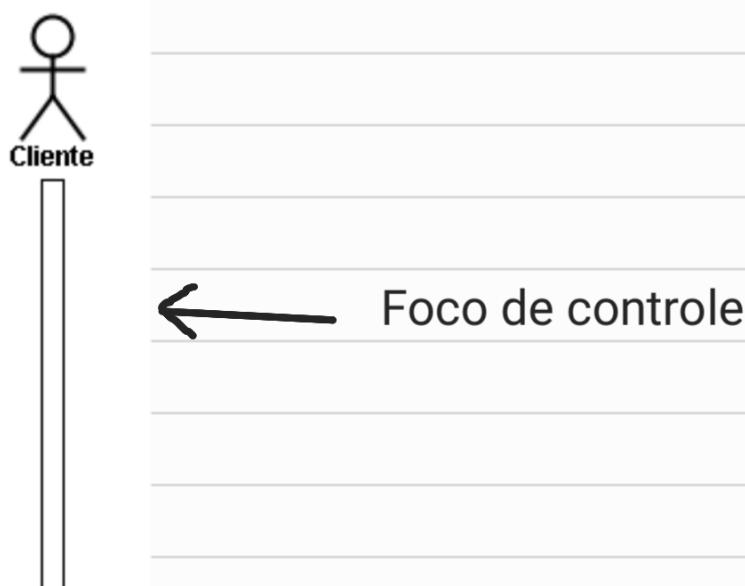
Linha de Vida

Representa o tempo em que um objeto existiu durante um processo através de linhas finas verticais tracejadas partindo do retângulo que representa o objeto



- **Foco de Controle ou Ativação**

Indica os períodos em que um determinado objeto está participando ativamente do processo, ou seja, identifica os momentos em que um objeto está executando um ou mais métodos utilizados em um processo específico e é representado por uma linha mais grossa



- **Mensagens ou Estímulos**

Demonstra a ocorrência de eventos, que normalmente forçam a chamada de um método em algum dos objetos envolvidos no processo.

Pode ocorrem, no entanto, de uma mensagem representar simplesmente a comunicação entre dois atores, o que, nesse caso não dispara nenhum método.

Um diagrama de sequência em geral é iniciado por um evento externo , causado por um ator, o que acarreta o disparo de um método em um dos objetos que inicia o processo.

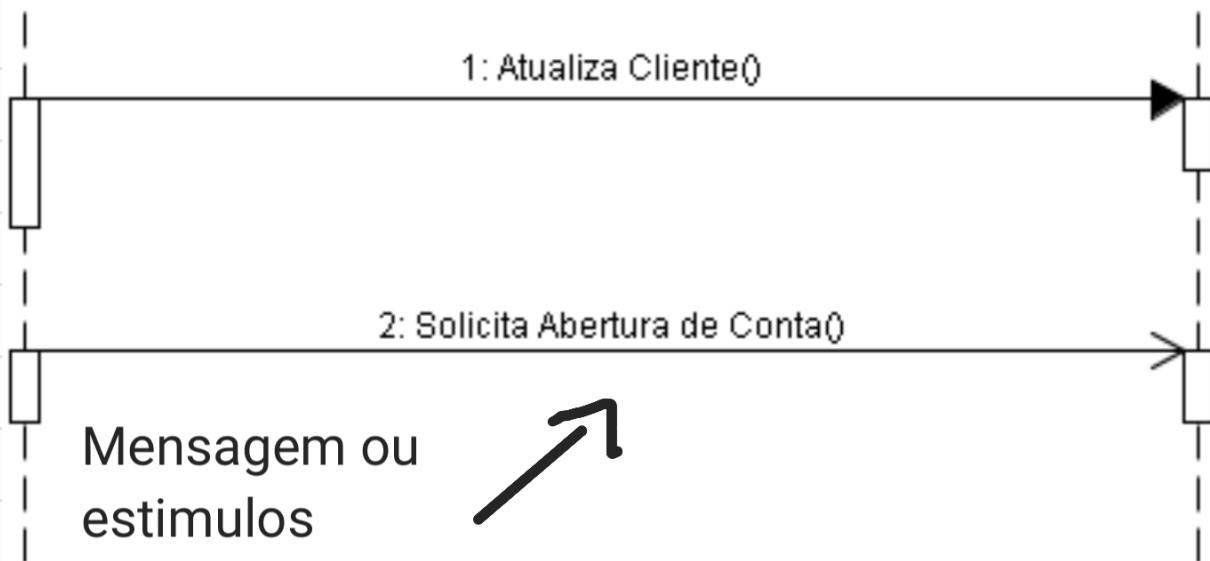
As mensagens podem ser disparadas entre:

Um ator e outro ator, o que não é muito comum , mas pode permitir uma compreensão melhor do processo como um todo

Um ator e um objeto, onde um ator produz em evento que força o disparo de um método em um objeto.

Um objeto e um objeto, o que constitui a ocorrência mais comum de mensagens, onde um objeto transmite uma mensagem a outro solicitando a execução de um método. Um objeto pode inclusive enviar uma mensagem para si mesmo, disparando um método em si próprio o que é conhecido como auto-chamada.

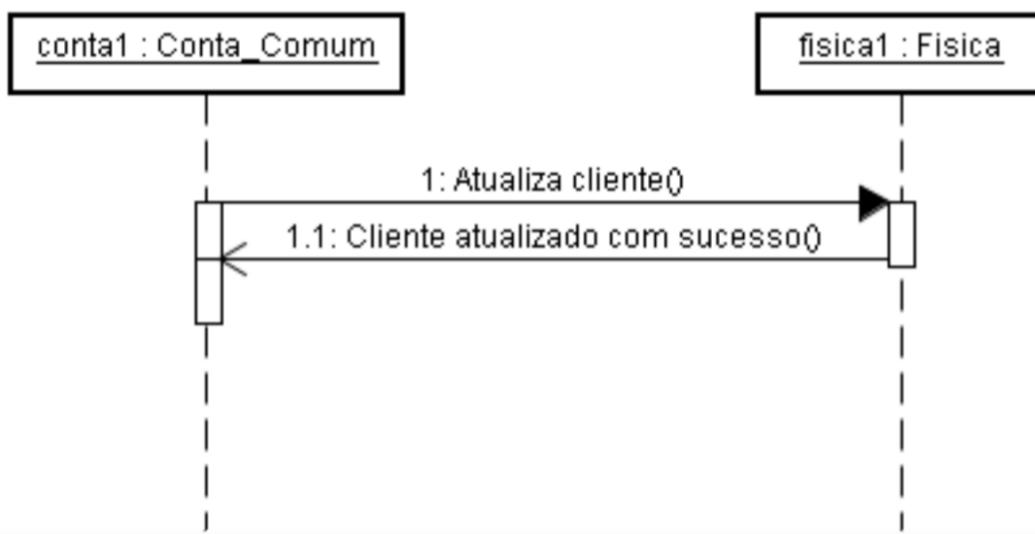
Um objeto e um ator, normalmente isso somente ocorre quando um objeto envia uma mensagem de retorno em resposta a chamada de um método solicitado, contendo seus resultados.



- **Mensagem de retorno**

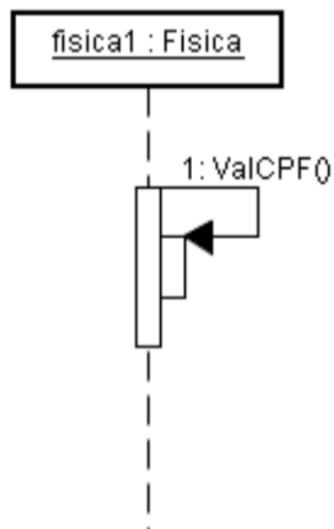
Este tipo de mensagem indica a resposta a uma mensagem para o objeto ou ator que a chamou.

Pode retornar informações específicas do método ou simplesmente um valor indicando se o método foi executado com sucesso ou não.



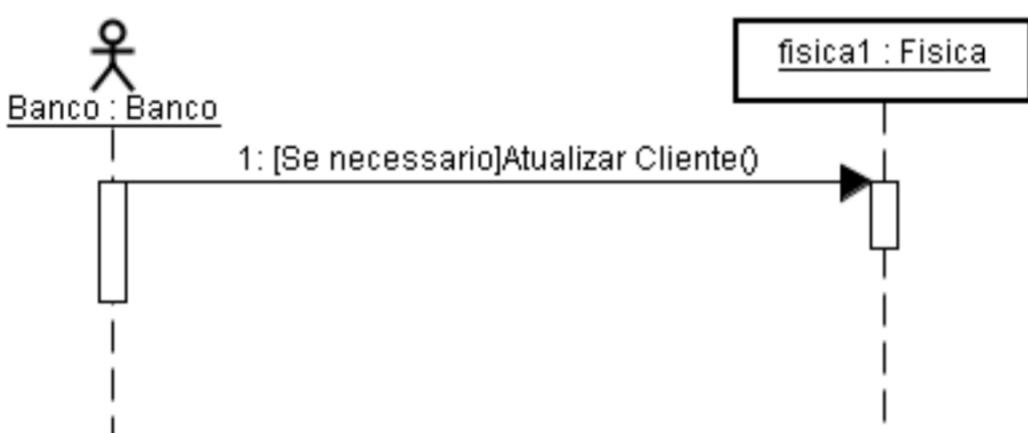
- **Autochamadas**

São mensagens que um objeto envia para si mesmo



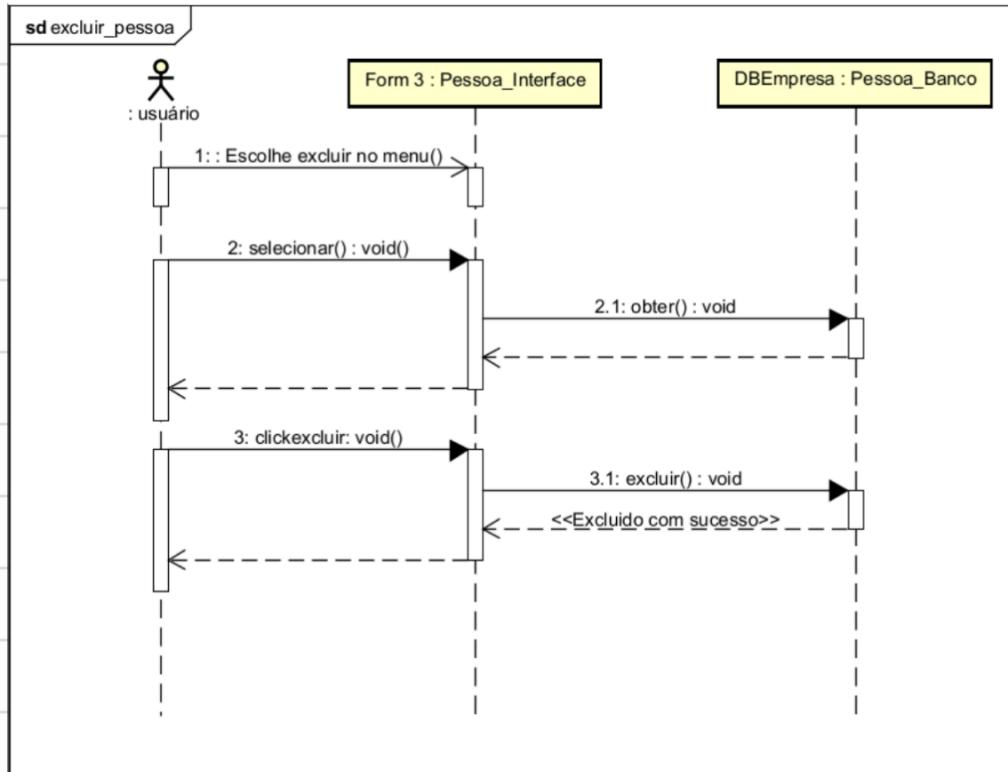
- **Condições**

Indicam que uma mensagem só poderá ser enviada a um objeto se uma determinada condição for verdadeira. As condições são descritas normalmente entre colchetes “[]”.

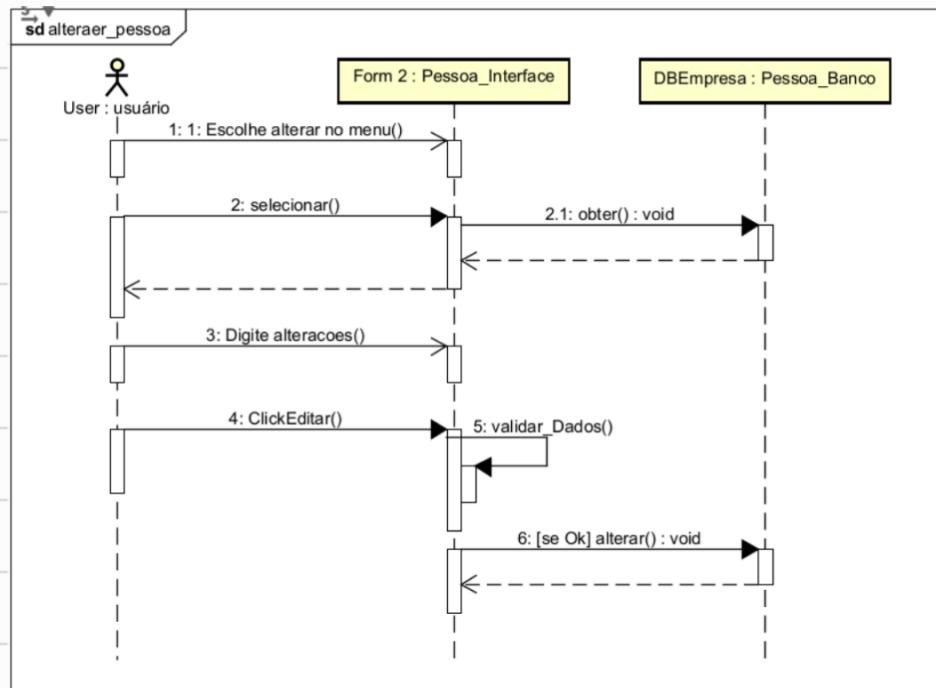


Exemplos:

Passo a passo de excluir



Passo a Passo para Alterar:



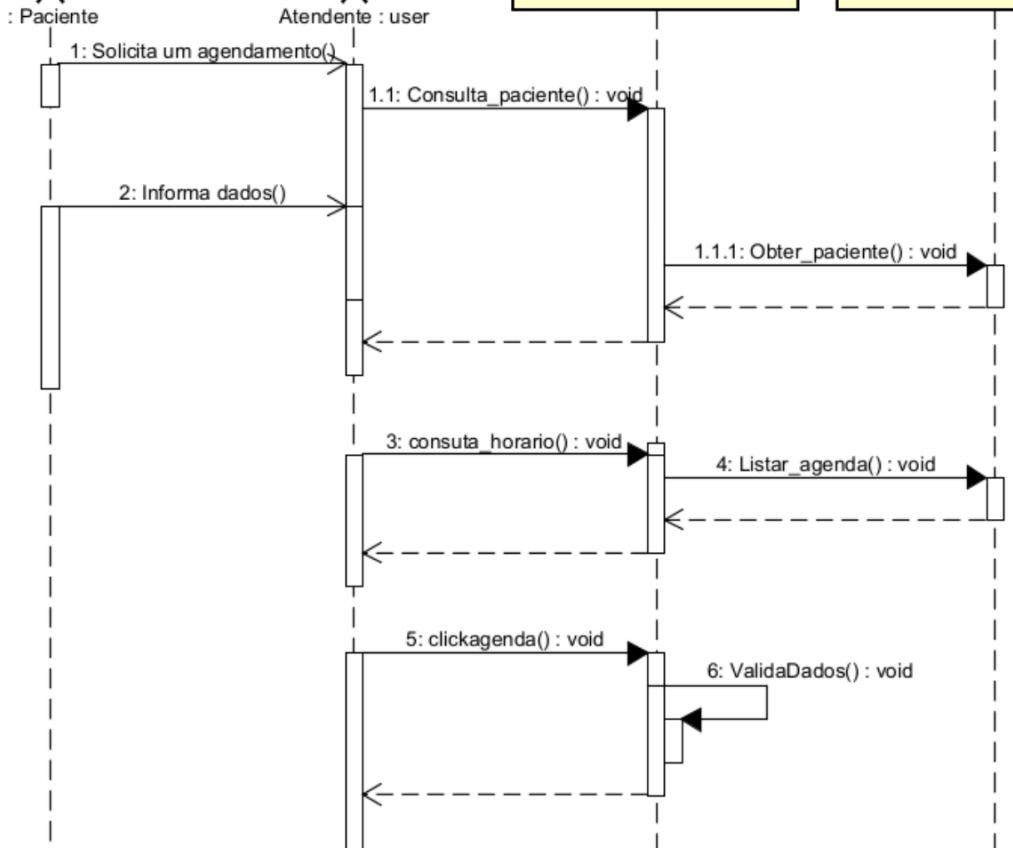
Usuário de negocio
não executa codigos
ele não mexe no sistema

Ações sincronas sempre podem ter retorno

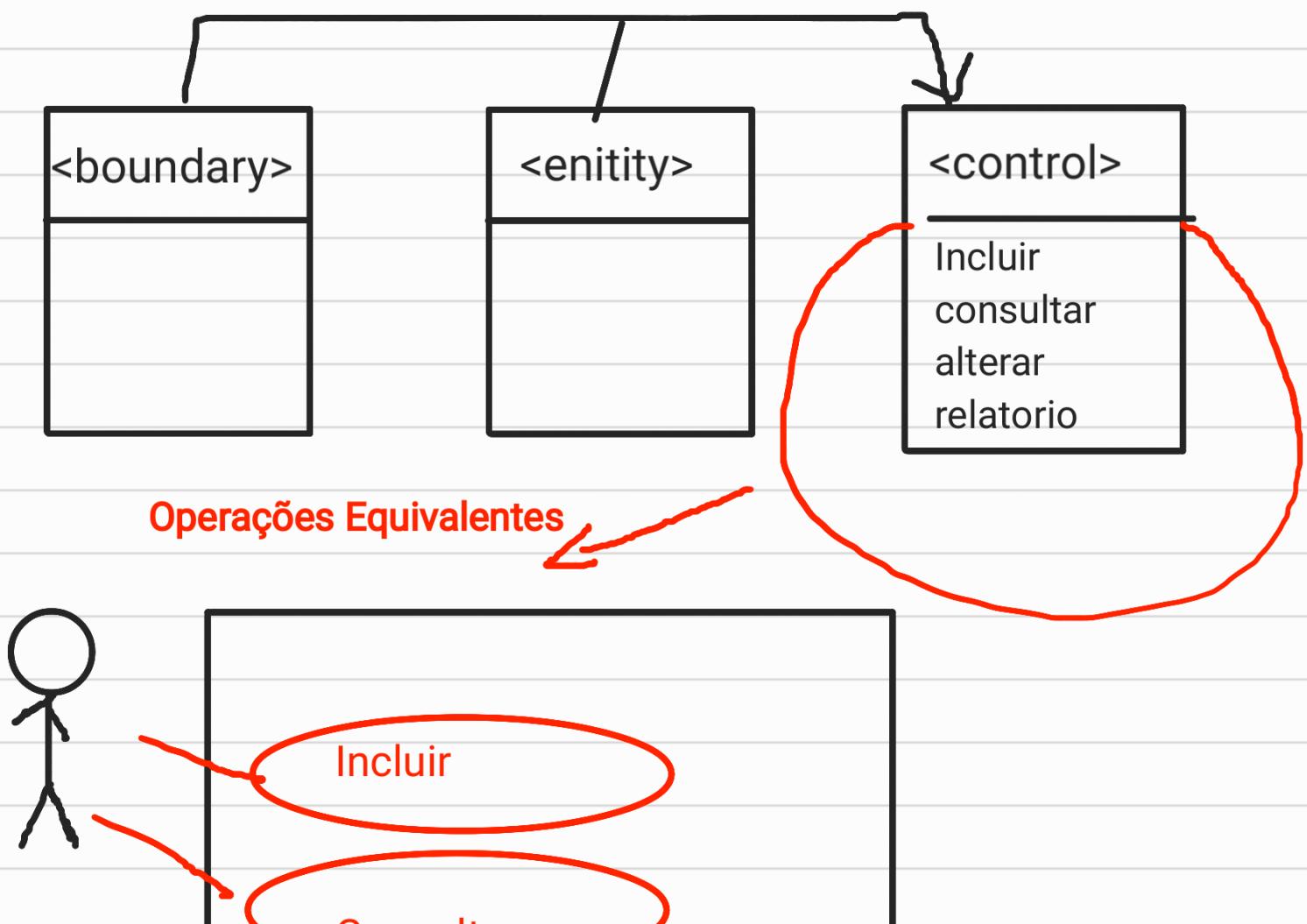
Reply(R) Retorno
nem sempre tem
mensagem de feedback

Form 1 : Paciente_Interface

BancoDados : Paciente_dados



O diagrama com esteriótipo também gera um diagrama de sequencia:



Consultar

alterar

relatório

Material 7: Diagrama de Atividade

Diagrama de atividades, aquele que lembra um fluxograma.

É um diagrama baseado em redes de Petri com maior ênfase ao nível de algoritmo da UML e um dos mais detalhistas.

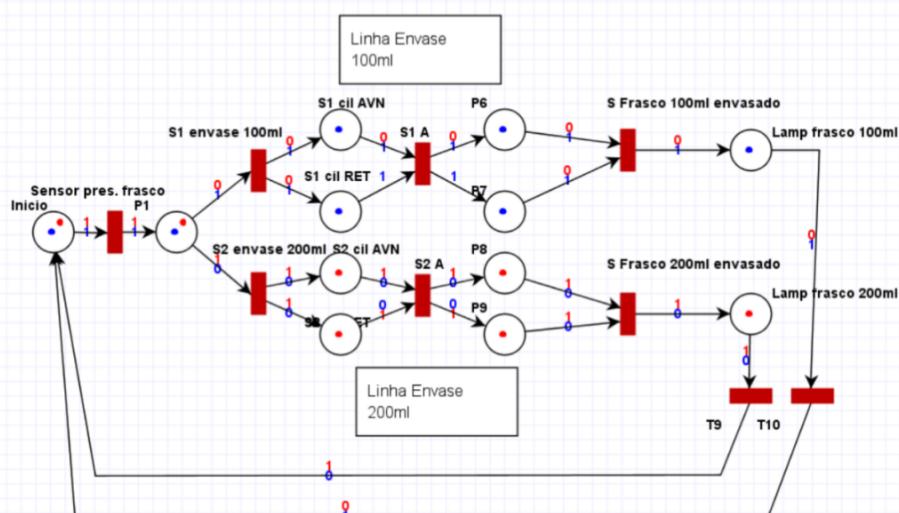


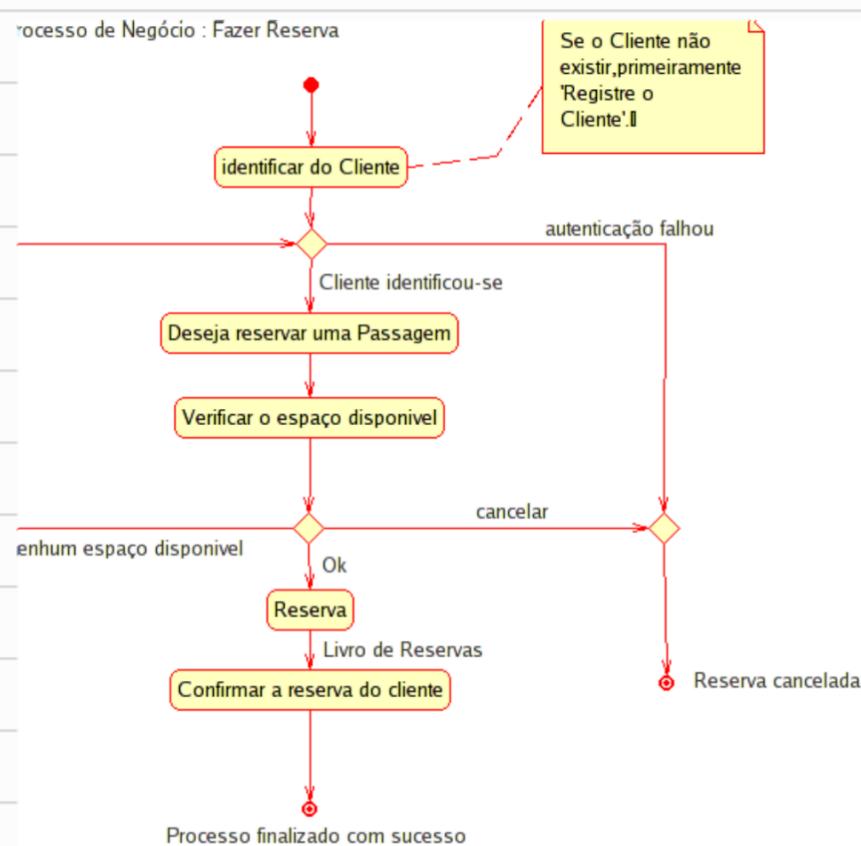
Diagrama de Petri

Apresenta muita semelhança com fluxogramas de lógica e programação sendo muito comum utilizar pseudocódigos ou mesmo linguagem de programação para representá-lo.

O diagrama pode ser usado para:

- Descrever atividades e fluxos de dados ou decisões entre atividades;

- Fornecer uma visualização ampla dos processos comerciais;
- Descrever as atividades que ocorrem em um caso de uso;
- Mostrar muitas atividades diferentes usando pequenos símbolos diferentes;
- Mostrar segmentos paralelos;



Elementos do diagrama atual:

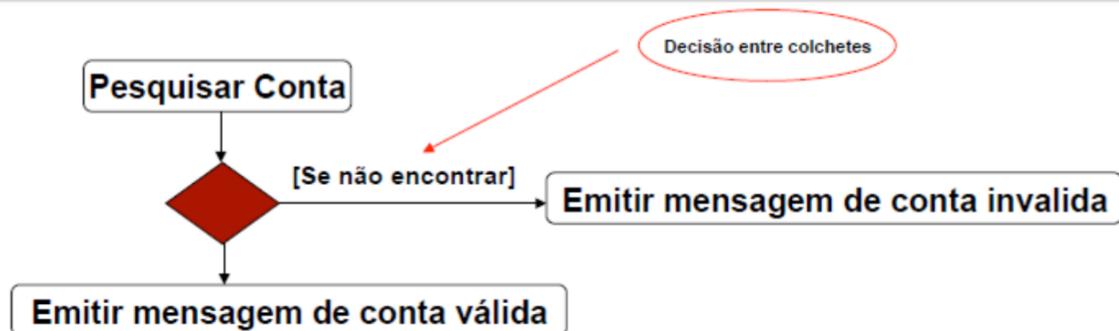
- **Estado de ação**

Representa a realização de uma ação dentro de um fluxo de controle, é atômico, ou seja, não pode ser decomposto em sub-estados.

Estado de Ação

- **Ponto de Decisão**

Representa um ponto do fluxo de controle onde deve ser realizado um teste, uma tomada de decisão



- **Início**

Inicia um diagrama de atividades

- **Fim**

Finaliza o diagrama, pode ter mais de um fim em todo diagrama



Exemplificando o diagrama

Descreve passo a passo, processo explicado.

Não é muito de TI, e sim de processo.

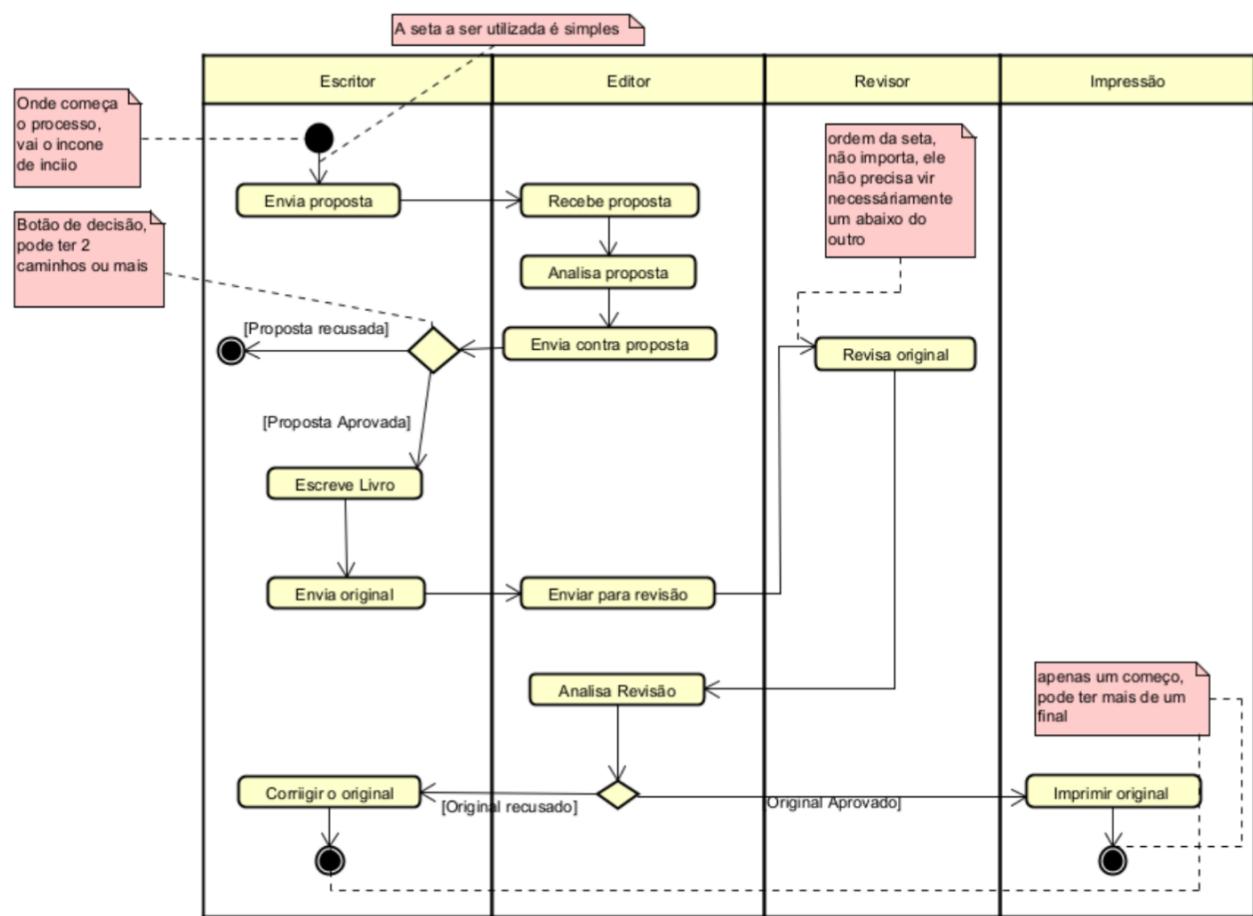
Raia vertical

Escrivtor	Editor	Revisor	Impressão
			○

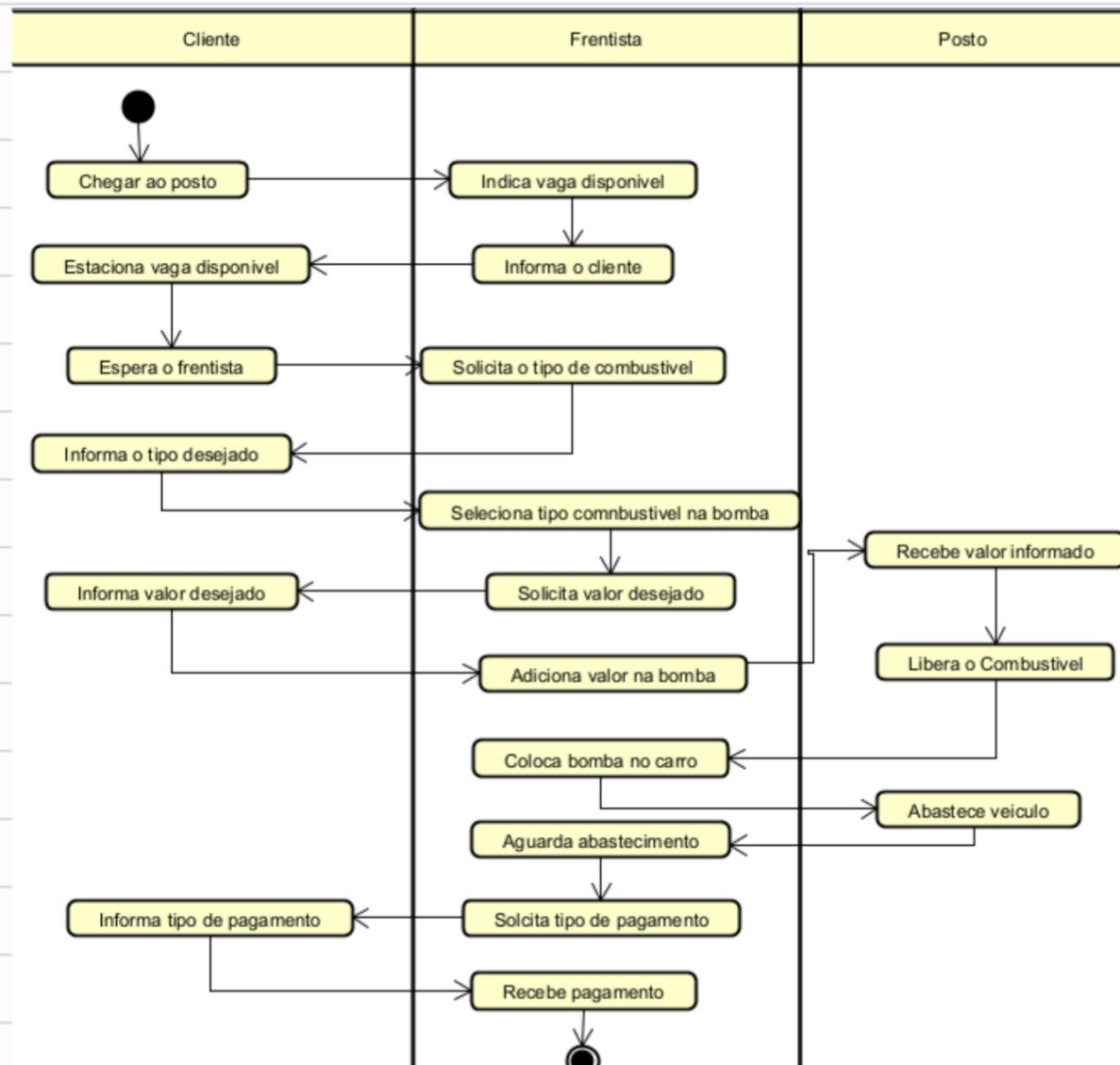
Demonstra de quem é a responsabilidade daquele processo.

Raia de Natação - swimlanes As raias de Natação são uma extensão do diagrama de atividades, onde procura-se identificar os diversos setores, departamentos ou mesmo os atores que interagem com um processo. São representadas por retângulos com divisões por zonas de influencias de um determinado setor sobre um determinado processo

Exemplo:

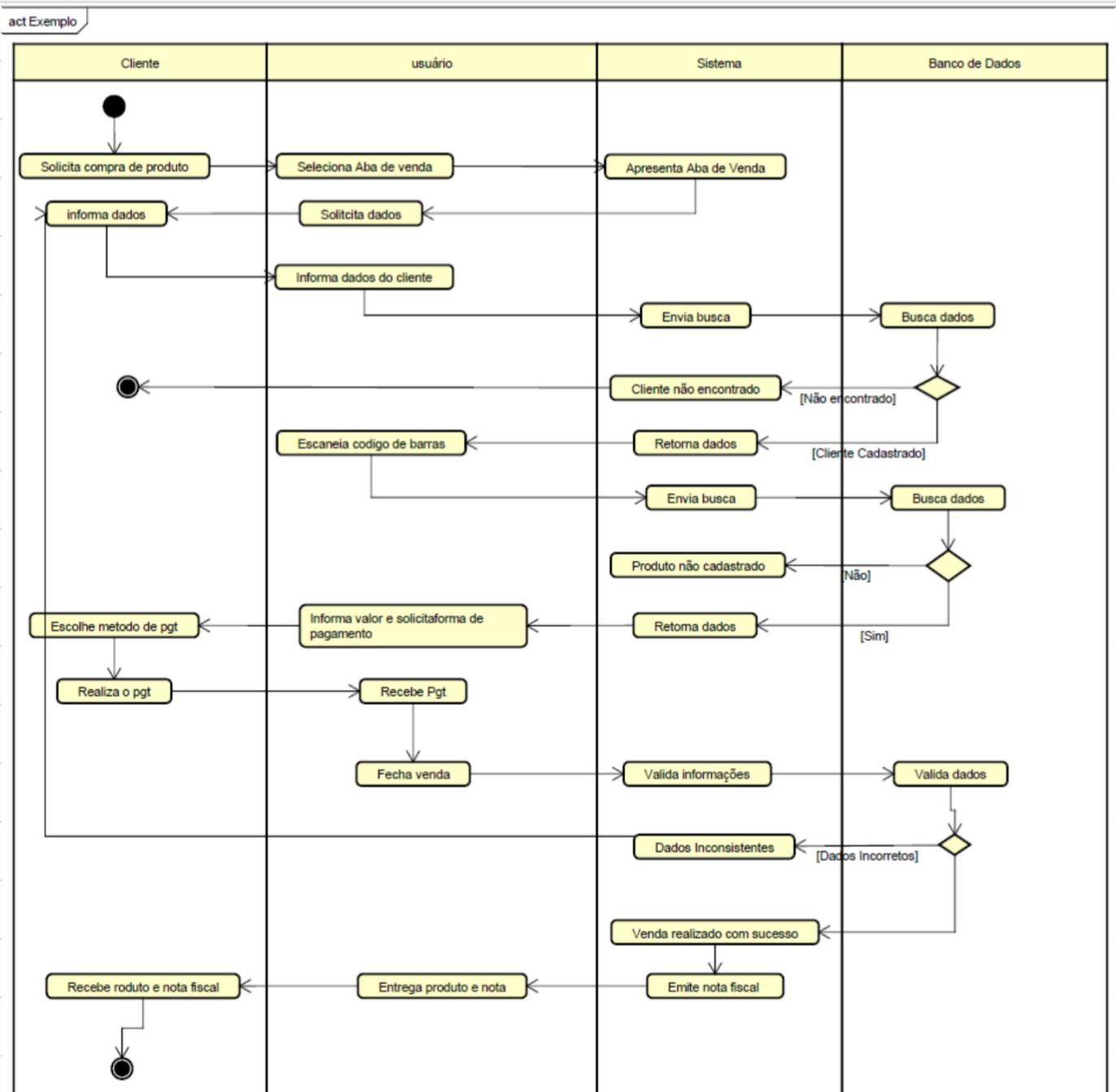


Exemplo de posto de gasolina, versão Adryelle



Exemplos Padrão no arquivo: Exemplos Padrão DA

O **Cliente** solicita a compra e um produto que ele gostou, o usuário/vendedor do **sistema** abre a venda, uma tela é apresentada para que o usuário informe o nome cliente que deve estar previamente cadastrado. Caso não encontre o cliente exibir a mensagem “Cliente não encontrado!”. Em seguida o **usuário/vendedor** escaneia o código de barras do produto e o **sistema** busca no banco de dados trazendo descrição e valor dele ou a mensagem “Produto não cadastrado”. Considere que os produtos são vendidos individualmente para efeitos fiscais. Em seguida o usuário/vendedor questiona o cliente sobre a forma de pagamento (dinheiro ou cartão) e efetua o recebimento e fecha a venda. O sistema por sua vez valida as informações do cliente, grava a venda e emite a mensagem “Venda realizada com sucesso!” ou caso haja algum problema “Dados inconsistentes!” e retorne ao recebimento. No final a nota fiscal deve ser emitida e entregue ao cliente junto ao produto adquirido.



Material 8: Diagrama de implementação

Diagramas Componentes : Linguagem

Diagrama de implantação: Infra

Diagramas Componentes : Linguagem

Diagrama de implantação: Infra



Diagrama de implementação

Diagramas de Implementação

Diagramas de implementação mostram aspectos de implementação física, incluindo a estrutura de componentes e a estrutura do sistema em tempo de execução (run time).

Eles são expressos em duas formas:

- Diagrama de Componentes
- Diagramas de Implantação

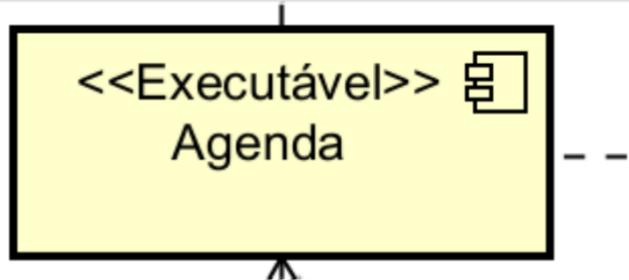
Diagrama de componentes:

Especifica um conjunto de construções que podem ser usadas para definir sistemas de software. O componente é identificado como uma unidade modular com interfaces bem definidas e substituíveis dentro de seu ambiente.

O objetivo do diagrama de componentes é o de mostrar caixas pretas. Essas caixas pretas, que são os componentes, devem especificar quais são suas interfaces, para que outros componentes possam acessar seus serviços sem que para isso precisem conhecer seu conteúdo.

Componentes:

Cada arquivo de compõe o sistema pode ser considerado um componente. São representados por um retângulo principal , contendo o nome do modulo que ele representa com dois retângulos menores sobressaindo-se à sua esquerda.



Vários estereótipos podem ser usados nesse diagrama para especificar o tipo desse componente.

- Executável (executable)
- Biblioteca (library)
- Tabela (table)
- Documento (document)
- Arquivo (file)

Estereótipos - Executável <<executavel>>

Determina que o componente em questão é um executável, ou seja, um arquivo já compilado para executar um conjunto de instruções

Estereótipos - Biblioteca <<biblioteca>>

Refere-se a bibliotecas contendo funções e sub rotinas que podem ser compartilhadas por diversos componentes executáveis podendo ser fornecidos pela própria linguagem ou por seus desenvolvedores.

Estereótipos - Tabela <<tabela>>

Refere-se a repositórios físicos de dados, onde os registros produzidos pelo sistema deverão ser armazenados.

Estereótipos - Documento <<documento>>

Este estereótipo faz referência a arquivos de texto utilizados pelo sistema por exemplo arquivos de help (ajuda).

Estereótipos - Arquivo <<arquivo>>

Refere-se a qualquer outro arquivo que componha o sistema, como arquivos de código-fonte dos módulos do sistema por exemplo.

Dependência

Um componente pode utilizar serviços ou depender de outros componentes do sistema. Isso é determinado por meio do relacionamento de dependência através de uma seta pontilhada.

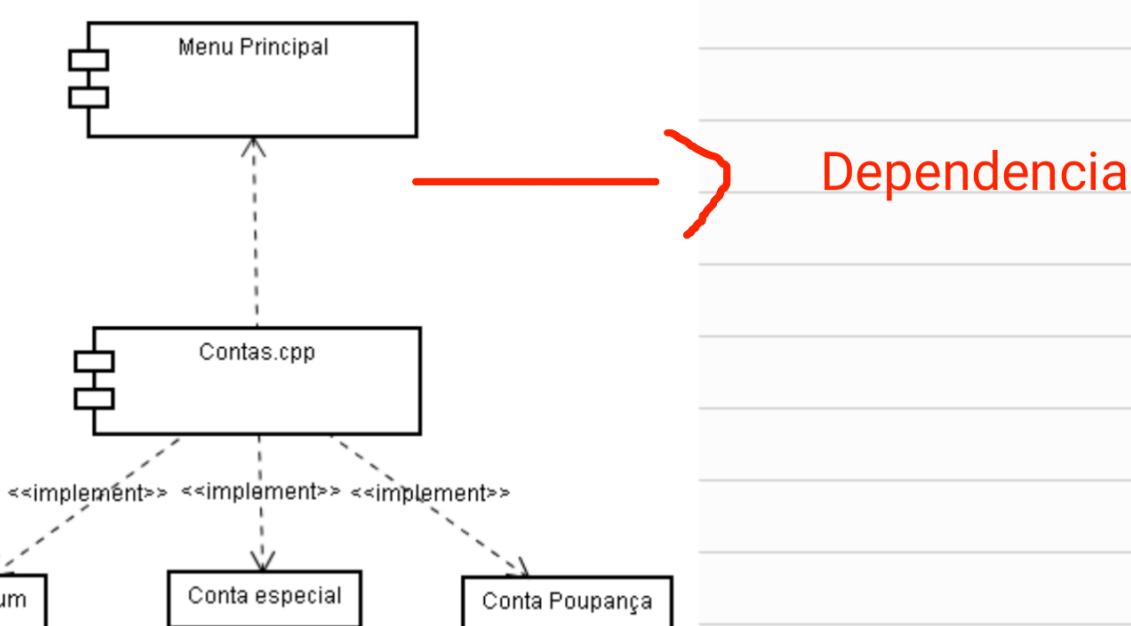
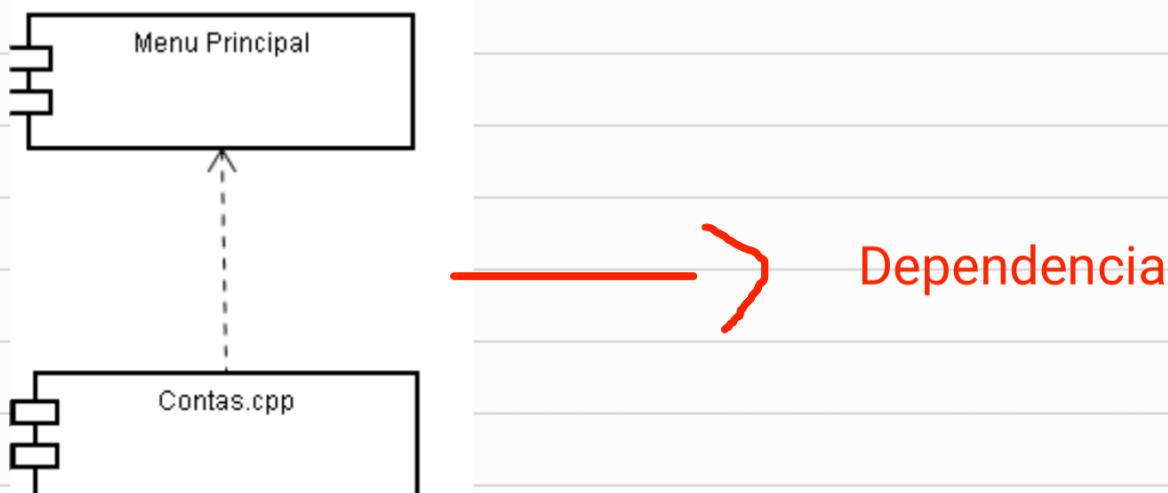




Diagrama de Implantação

- É o diagrama com a visão mais física da UML. Esse diagrama enfoca a questão da organização e da estrutura física na qual o software será implantado.
- Mostra como os equipamentos estarão conectados através dos protocolos de comunicação.
- Este diagrama só se faz necessário quando o sistema modelado for executado em múltipla máquinas sendo assim, não é requisitado quando o projeto for apenas para um equipamento.

Nós

- São componentes básicos do diagrama de implantação e representa uma máquina onde um ou mais módulos do sistema serão executados. É representado por um Cubo e o nome do item que ele representa

Caixa Eletronico

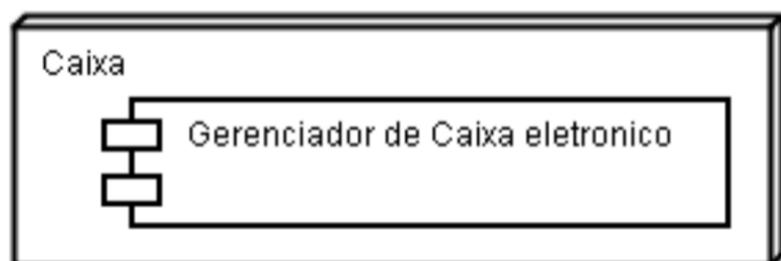
Associação

Os nós possuem ligações físicas que possibilitam trocar informações essas são chamadas de associações e são representadas por retas ligando um nó a outro.

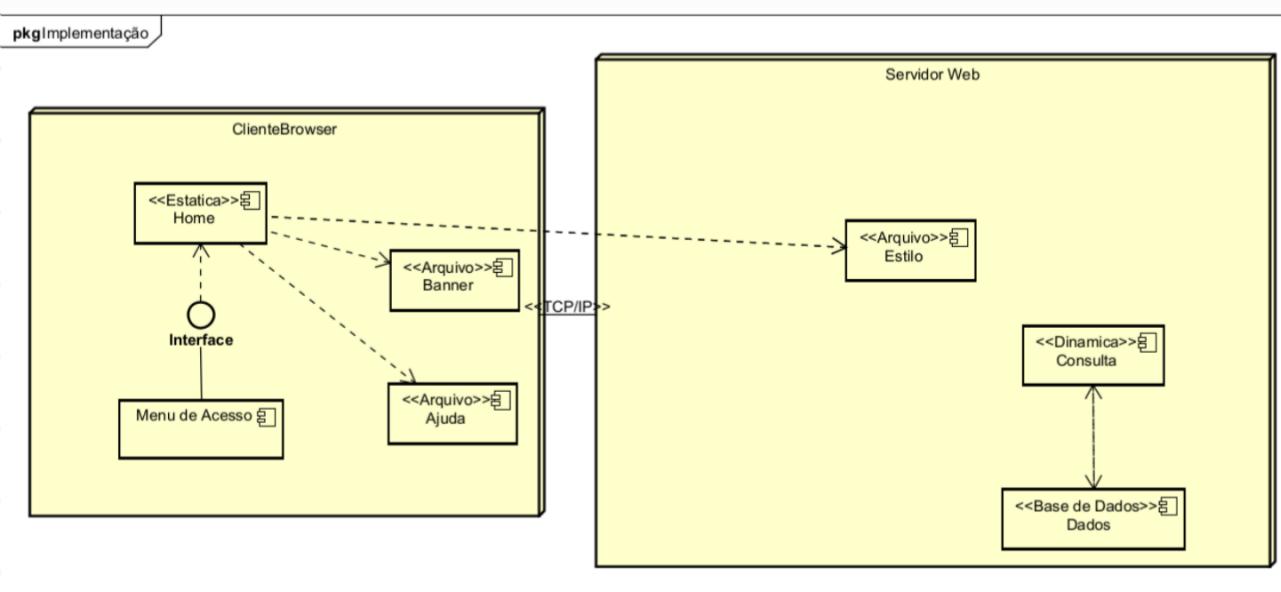


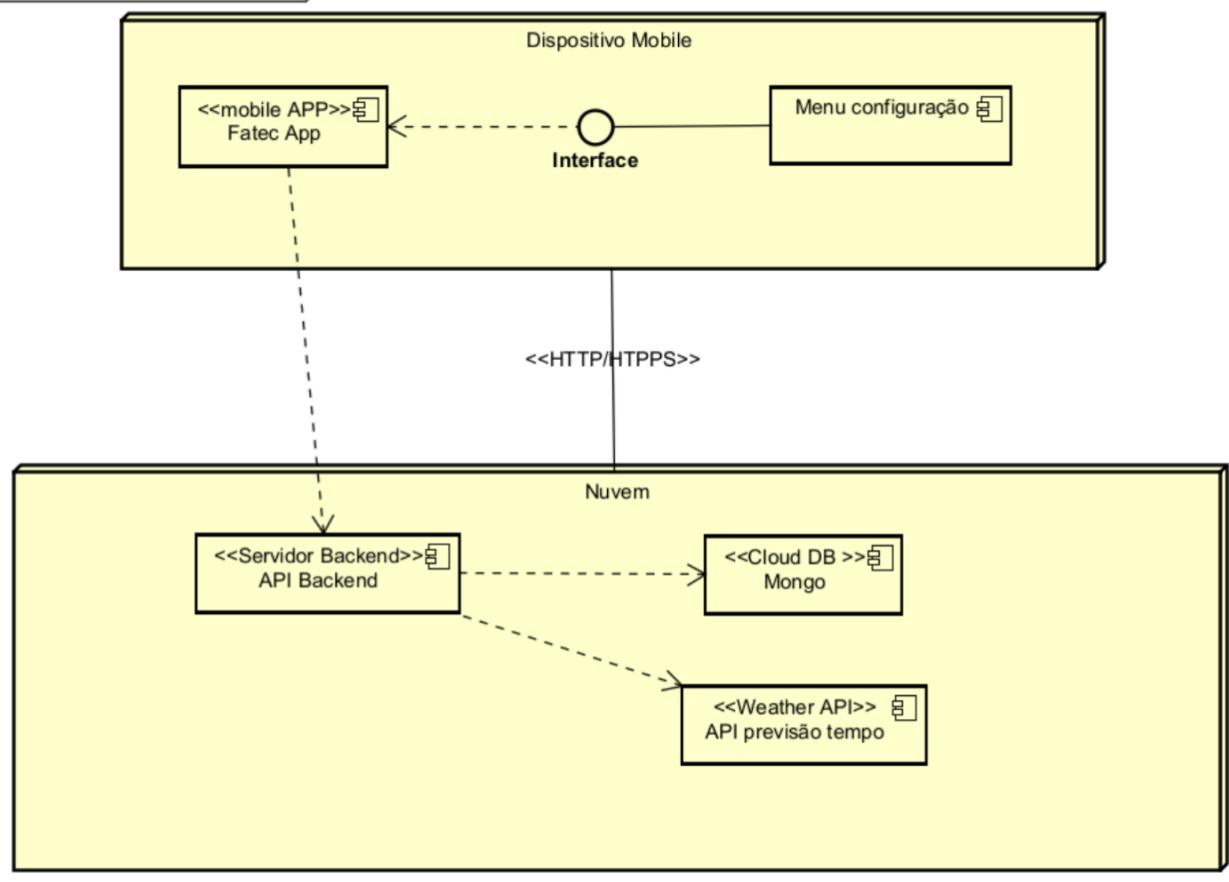
Nós com componentes

Os diagramas de implantação e de componentes podem ser combinados podendo ser modelados juntos, sendo comum identificar os componentes que são executados por um nó.



Exemplo





Link dos exercícios resolvidos: https://github.com/DryCaleffi/Engenharia_Soft_3
(OBS: estão em arquivos do Astah)

