**UML**

**UML:** Linguagem de Modelagem Unificada, é uma linguagem de modelagem que permite representar um sistema de forma padronizada.

* **Vantagem:** permite que desenvolvedores visualizem os produtos de seus trabalhos em diagramas padronizados e é a única que abarca análise, design, implementação e teste.

**Tipos de Diagramas UML:**

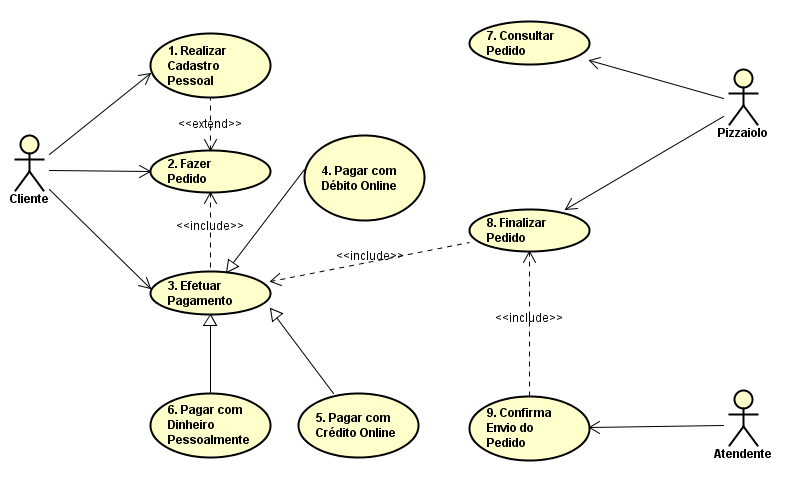
* **Estáticos:** Casos de uso; Classes; Objetos; Componentes; Implantação.
* **Dinâmicos:** Sequência; Colaborações; Estados; Atividades.

→ O modelo do sistema é representado pelos dois conjuntos de diagrama: estático e dinâmico.

→ Um diagrama apresenta uma representação parcial do sistema e ajuda a compreender a arquitetura do sistema em desenvolvimento.

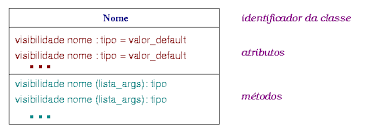
**Diagrama de casos de usos:** mais geral e informal, utilizado, normalmente, nas fases de levantamento e análise de requisitos do sistema. É simples enquanto linguagem e de fácil entendimento. Apresenta uma situação que demonstra as funcionalidades do sistema do ponto de vista do usuário.

* **Objetivo:** auxiliar a comunicação entre os analistas e o cliente. Descreve um cenário que mostra as funcionalidades do sistema do ponto de vista do usuário. O cliente deve ver no diagrama de Casos de Uso as principais funcionalidades de seu sistema.
* **Notação:** atores; casos de uso; relacionamentos entre os elementos (associações, generalizações, extends e includs.



**Diagrama de Classes:** define a estrutura das classes utilizadas pelo sistema indicando os atributos e métodos de cada classe, e designa como as classes se relacionam e trocam informações.

* **Objetivo:** Descrever os vários tipos de objetos no sistema e o relacionamento entre eles. Um diagrama de classes contém: Classes, relacionamentos e notações.
  + **Classe:** nomeadas de acordo com o domínio do problema. São representadas por retângulos com nome, atributos e métodos. **Atributos:** representam o conjunto de características dos objetos da classe e sua visibilidade pode ser: Pública (+): visível de qualquer classe de qualquer pacote; Protegido (#): visível apenas para classes do mesmo pacote; Privado (-): visível somente para a classe. **Métodos:** Representam o conjunto de operações que a classe fornece e seus tipos de visibilidades são idênticas ao dos atributos (público, protegido e privado).



* **Relacionamentos:** possuem nome, sentido de leitura, navegabilidade, multiplicidade, tipo e papéis. **Nome:** descrição dada ao relacionamento. **Navegabilidade:** é indicada por uma seta ao fim do relacionamento. **Multiplicidade**: 0..1 – zero ou uma instância; 0..\* ou \* -- sem limite para o número de instâncias; 1 – apenas uma instância ;1..\* - ao menos uma instância. **Tipos:** Associação - indica que os objetos de uma classe estão vinculados a objetos de outra, é representada por uma linha conectando as classes e contém os subtipos agregação (utilizada para indicar “todo-parte”) e composição (objetos “parte” só podem pertencer a um único objeto “todo” e o tempo de vida é coincidente entre si); Generalização – relacionamento entre itens gerais (superclasses) e específicos (subclasses); Dependência – alteração de um objeto pode afetar outro objeto.
* **Perspectivas:** um diagrama de classes pode oferecer três perspectivas, sendo cada uma para um usuário diferente:conceitos e entidades, classes, classes de software. **Conceitos e entidades:** representa os conceitos de domínio em estudo e tem a perspectiva do cliente. **Classes:** o foco é nas principais interfaces da arquitetura, nos principais métodos e tem a ótica das pessoas que não precisam detalhes de desenvolvimento. **Classes de software:** aborda detalhes de implementação e tem o prisma do time de desenvolvimento.

**Diagrama de Objeto:** é um complemento do diagrama de classes e mostra os valores armazenados no objeto num momento da execução do processo.

**Diagrama de Componentes:** consiste num conjunto de artefatos físicos em formato digital, arquivos de código ou arquivos de documentos relativos ao negócio. Um componente é uma peça básica na implementação de um sistema. Tipos: componentes de instalação, trabalho e execução. É associado à linguagem de desenvolvimento do sistema, representa os componentes do sistema em implementação e, determina, como agirão e quais estarão estruturados.

**Diagramas de Implantação:** define toda necessidade de hardware para o sistema ser implementado.

**Diagrama de Sequência:** diagrama comportamental voltado para a ordem temporal em que mensagens são trocadas entre objetos de um processo. É baseado em caso de uso e apoiado no de classes, a fim de definir objetos e classes de um processo. Elementos encontrados: linhas verticais, linhas horizontais ou diagonais, condição, mensagens de retorno.

* **Linhas verticais:** representa o tempo de vida de um objeto, são preenchidas com barras verticais indicando quando um objeto passou a existir e um ‘X’ quando deixou de existir.
* **Linhas horizontais ou diagonais:** representa mensagens trocadas entre os objetos, contém um rótulo com o nome da mensagem.
* **Condição:** é representada por uma mensagem cujo rótulo é envolvido por colchetes.
* **Mensagens de retorno:** representadas por linhas horizontais tracejadas e só deve ser mostrada quando for fundamental para a clareza do diagrama.

**Diagrama de colaborações (comunicações):** é associado ao diagrama de sequência e apresenta mensagens como o mesmo, porém é voltado para a vinculação dos elementos do diagrama e as mensagens trocadas entre si no processo, e não ao tempo.

**Diagrama de Estados:** o objeto possui comportamento e estado que depende da atividade processada no momento. Apresenta os possíveis estados de um objeto e as transações responsáveis pelas mudanças de estado. Pode ser baseado em um caso de uso e pode acompanhar os estados de outros elementos, além de mostrar o comportamento de uma parte do sistema.

**Diagrama de Atividade:** aborda a descrição dos passos a serem percorridos para a conclusão de uma atividade, foca na representação do fluxo de controle da mesma. Podem manipular processos paralelos. Elementos mais comuns: activity, partition, decision, initial, final, fork e join.

* **Activity:** é a atividade em si
* **Partition:** ilustram fronteiras conforme o nível de detalhamento e enfoque, é chamado de “Raia”.
* **Decision:** representa uma decisão que pode desviar o fluxo ilustrado no diagrama.
* **Initial:** é o primeiro elemento do diagrama e define o início do fluxo.
* **Final:** é o último elemento do diagrama e define o término do fluxo.
* **Fork:** divide o fluxo em mais de uma direção.
* **Join:** faz a união de várias direções do fluxo em uma só direção.