APS

Introdução

O desenvolvimento de software é uma atividade de crescente importância na sociedade contemporânea. A utilização de computadores nas mais diversas áreas do conhecimento humano tem gerado uma crescente demando por soluções computadorizadas.

Contextualização – Marcos Históricos

Máquina a Vapor: Inventada (1712); Aperfeiçoada (1766 - '69 - '82)
Mecanização da Indústria Têxtil: Tear mecânico (1722); Máquina de fiar (1764)
Aço (1856 e 1885 – Liga)
Locomotiva a Vapor: Rede de transporte (1830)
Máquina de Costura (1851)
Motor a combustão interna: Primeiro eficiente (1876); Produção automobilística em massa (1896)
Energia nuclear (1943)
Uso industrial/Comercial da Eletricidade
Computadores Eletrônicos (1946)
Transistor (1948)

O que é software?

Programas de computador; Entidades abstratas; Ferramentas que: exploram os recursos de hardware, executam determinadas tarefas, resolvem problemas e tornam o computador operacional.

Segundo Pressman, software é...

É o Produto que os profissionais de software constroem e mantem ao longo do tempo. Abrange programas que executam em qualquer computador, conteúdo que é apresentado ao programa a ser executado e a documentação impressa ou virtual.

Panorama do SW

O que é? É o produto que os profissionais de software constroem e mantêm ao longo do tempo.
Quem faz? Engenheiros de Software constroem e mantêm e todas as pessoas que usam de maneira direita e indireta.
Por que é importante? Porque afeta todos os aspectos de nossas vidas e difundiu-se no comércio, cultura e no cotidiano.

Características do software

SW não é um elemento físico, é um elemento lógico (não tem propriedades físicas);

O software não se desgasta com o uso, mas deteriora-se;

Não há peças de reserva – manutenção, correção e aperfeiçoamento;

A maioria dos softwares é feita sob medida e não montada a partir de componentes existentes – reusabilidade.

Atributos essenciais de um bom software

Manutenibilidade = o software deve ser escrito de forma que possa evoluir para atender às necessidades dos clientes;
Confiança e proteção = confiabilidade, proteção e segurança devem fazer parte de um software. Um SW confiável não deve causar prejuízos físicos ou econômicos no caso de falha do sistema e usuários maliciosos não devem ser capazes de acessar ou prejudicar o sistema;
Eficiência = o software não deve desperdiçar os recursos do sistema, como memória e ciclos do processador, portanto eficiência inclui capacidade de resposta, tempo de processamento, uso de memória etc.
Aceitabilidade = ele deve ser aceitável para o tipo de usuário para o qual foi projetado, isso significa que deve ser compreensível, usável, e compatível com outros sistemas usados por ele.

Qualidade de software - exemplo

Correto: a loja não pode deixar de cobrar por produtos comprados pelo consumidor.
Robusto: a loja não pode parar de vender.
Eficiente: o consumidor não pode esperar; a empresa quer investir em poucos recursos computacionais.
Amigável e fácil de usar: a empresa quer investir pouco em treinamento.
Reutilizável: várias empresas precisam utilizar partes de um mesmo sistema.
Aberto, compatível, de fácil integração com outros sistemas: a empresa já tem controle de estoque, fidelização etc.
Portável e independente de plataforma (hardware e software): a empresa opta por uma determinada plataforma.
Baixo custo de instalação e atualização: a empresa tem um grande número de PDV's (ponto de venda).

Atividades do processo de desenvolvimento de software

Especificação de software = onde os clientes e engenheiros definem o software que deve ser produzido e as restrições sobre o seu funcionamento.
Desenvolvimento de software = o software é projetado e programado.
Validação = onde o software é verificado para garantir se as necessidades dos clientes foram atendidas.
Evolução = onde ocorrem modificações no software para refletir mudanças de requisitos do cliente e do mercado.

Questões que afetam a maioria dos SW's

Heterogeneidade: cada vez mais, os sistemas são necessários para operar como sistemas distribuídos através de redes que incluem diferentes tipos de computadores e dispositivos móveis.
Mudança de negócio e social: negócio e sociedade estão mudando com uma rapidez incrível, na medida em que as economias emergentes se desenvolvem e as novas tecnologias se tornam disponíveis.
Segurança e confiança: como o SW está entrelaçado com todos os aspectos de nossas vidas, é essencial que possamos confiar neles.

Histórico dos Softwares

Anos 1940 = Iniciou a evolução dos sistemas computadorizados, a maior parte dos esforços, e custos, era direcionada para o desenvolvimento do hardware, devido principalmente das limitações e dificuldades encontradas na época.
Anos 1950 - 1965 – Primeira Era = Softwares customizados e processamento Batch.
Anos 1965 - 1975 – Segunda Era = Surgimento dos comutadores multiusuários e processamento em tempo real.
Anos 1975 – 1985 – Terceira Era = Diminuição dos custos dos computadores, hardware de baixo custo e demanda crescente por softwares mais complexos.
Processamento distribuído.
Anos 1985 – Quarta Era = Sistemas especialistas, computação paralela, necessidade de a informação estar disponível em curto espaço de tempo para muitas pessoas (internet).

Aplicação de software

Software Básico

É uma coleção de programas escritos para dar apoio a outros programas. Exemplos: compiladores, editores e utilitários para gerenciamento de arquivos.

Software de tempo real

Software que monitora/analisa/controla eventos do mundo real, mas com tempo de resposta em tempo real (milissegundo a 1 minuto).

Software de tempo real

São aplicações que processam e disponibilizam informações para apoiar a tomada de decisão pelos gestores: Sistemas de folhas de pagamento, contas a pagar/receber, controle de estoque e transações em ponto de venda, por exemplo.

Software cientifico e de engenharia

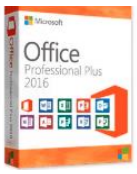
São aqueles que auxiliam as aplicações cientificas, tem como características algoritmos de intenso processamento de números e cálculos. São exemplos a Astronomia, Vulcanologia, análise da fadiga mecânica de automóveis e Biologia Molecular.

Software embarcado

Reside na memória de leitura e é usado para controlar produtos e sistemas para os mercados industriais e de consumo. Executam funções limitadas e particulares.

Software de computador pessoal

Tornou-se muito procurado e continua a representar os mais inovadores projetos de interface com seres humanos de toda a indústria de software.



Software de inteligência artificial

Faz uso de algoritmos não numéricos para resolver problemas complexos que não sejam favoráveis à computação ou à análise direta.

Crise de Software

25% dos projetos são cancelados; O tempo de desenvolvimento é bem maior do que o estimado; 75% dos sistemas não funcionam como planejado; A manutenção e reutilização são difíceis e custosas; Os problemas são proporcionais a complexidade dos sistemas; Perda do conhecimento quando as pessoas saem.

Causas da crise de software

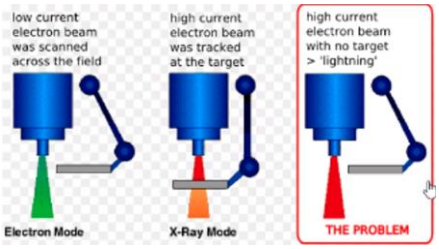
Essenciais: complexidade dos sistemas e dificuldade de formalização;
Acidentes: má qualidade dos métodos, linguagens, ferramentas, processos e modelos de ciclo de vida e a falta de qualificação técnica.

Importante!

Estimativas de prazo e de custo; Produtividade das pessoas; Qualidade de software; Software difícil de manter.

Therac-25

Equipamento de radioterapia. Entre 1985 e 1987, ocorreram 6 acidentes causando mortes por overdoses de radiação – falhas por falta de testes integrados e falta de documentação.



Aeroporto Internacional de Denver

Custo do projeto US$ 4,9 bi.; 100 mil passageiros por dia; 1.200 voos; 53 milhas quadradas; 94 portões de embarque/desembarque; 6 pistas de pouso/decolagem....
Erros no sistema automático de transporte de bagagens: atraso na abertura do aeroporto com custo estimado em US$ 360 milhões e US$ 86 milhões para corrigir o sistema.

Ariane 5 – Projeto Espacial Europeu

US$ 8 bilhões, 10 anos e garantiria a supremacia europeia no espaço.
O que aconteceu? Explosão após 40 segundos da decolagem = o veículo detonou suas cargas explosivas de autodestruição, fazendo o foguete perder o controle de direção.
Culpa? Os computadores principais e de backup deram shutdown ao mesmo tempo.

```
strict precondition 1:
{       Set."x"=FLPT and Set."y"=INT16
        and -32768 <= x <= +32767
}
program code:
   y := int(x);
postcondition:
 {Set."x"=FLPT and Set."y"=INT16 and y=int(x)}
```

Ocorreu um run time error em um programa que convertia um valor em ponto flutuante para o inteiro de 16 bites (recebeu um valor fora da faixa permitida).

Engenharia de Software

É uma forma de engenharia que aplica os princípios da Ciência da Computação e Matemática para alcançar soluções com melhor custo-benefício para problemas de software.
Os fundamentos científicos para a engenharia de software envolvem o uso de modelos abstratos e preciso que permitem ao engenheiro especificar, projetar, implementar e manter sistemas de software, avaliando e garantindo suas qualidades.

Mitos de Software

Administrativo

Já temos um manual repleto de padrões e procedimentos para a construção de software. Isso não oferecerá ao meu pessoal tudo o que eles precisam saber?
Realidade = Será que o Manuel é usado? Os profissionais sabem que ele existe? Ele reflete a prática moderna de desenvolvimento de software? Ele é completo?

Administrativo II

Meu pessoal tem ferramentas de desenvolvimento de software de última geração. Afinal compramos os mais novos computadores.
Realidade = É preciso muito mais do que os mais recentes computadores para se fazer um desenvolvimento de software de alta qualidade.

Administrativo III

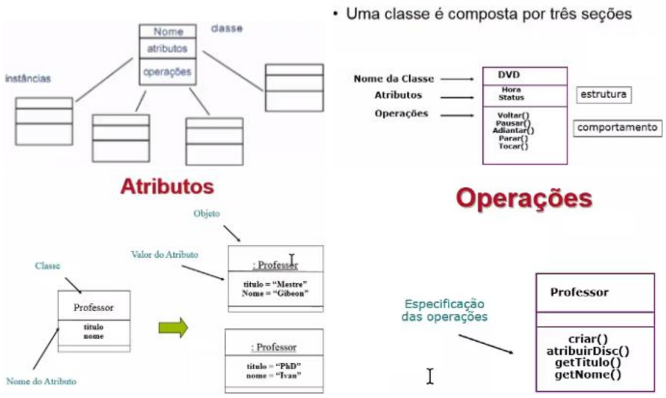
Se nós estamos atrasados nos prazos, podemos adicionar mais programadores e tirar o atraso.
Realidade = O desenvolvimento de software não é um processo mecânico igual à manufatura. Acrescentar pessoas em um projeto torna-o ainda mais atrasado. Pessoas podem ser acrescentadas, mas somente de uma forma planejada.

Orientação a Objeto

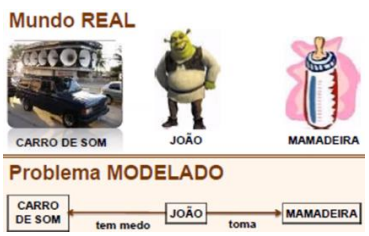
Conceitos

Classes e Instâncias
- Classe é uma categoria geral de objeto que descreve um conjunto de objetos específicos similares que são suas instâncias.
- É um elemento em OO que descreve o que seus objetos têm e fazem.
- Servem de base/molde para a criação de novos objetos que serão as instâncias.
- Instância é um sinônimo para o termo objeto.

Uma classe está associada com as suas instâncias por meio da relação de instanciação indicando que os atributos e as operações definidos na classe são repassados para suas instâncias.



Abstração



Domínio e Aplicação

Domínio Bancário:
Conta do José, Maria e João
Cliente José, Maria e João
Funcionário Jonas, Rafael e Marcelo

Definição de objeto

É uma entidade real ou abstrata, que modela um conceito presente na realidade humana, ocupando espaço físico ou lógico.
Um objeto é uma pessoa, um lugar, é a base para todos os outros conceitos da orientação a objetos.
Facilita a compreensão do mundo real e oferece uma base real para implementação em computador.
Um objeto denota uma entidade de natureza física, conceitual ou de software:
Entidades físicas = um carro, uma pessoa, um livro;
Entidade conceitual = um DER de um sistema;
Entidade de software = um radiobutton em uma página web.

Classes

Conta
numero saldo limite
saca() deposita() imprimeExtrato()

Uma classe é o projeto de um objeto.
Uma classe representa uma categoria e os objetos são membros dessa categoria.
Classe é a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo.

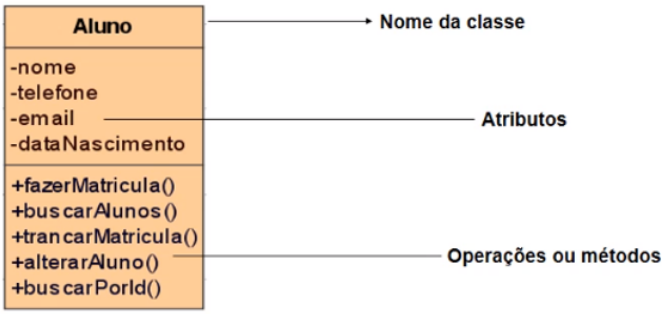
Uma classe é considerada uma fábrica de instâncias que inclui atributos e operações dessas instâncias.
É importante evitar a criação de classes que tentem abranger tudo (ou várias coisas).

- Classe Pessoa (grupo de objetos similares que compartilham atributos e comportamentos)

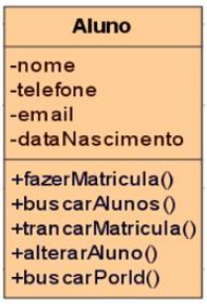
Analogia

Um objeto é como se fosse uma casa ou um prédio;
Para ser construído, precisa de um espaço físico (No caso dos objetos, esse espaço é algum trecho vago da memória do computador que executa a aplicação, e no caso das casas e dos prédios, o espaço físico é algum terreno vazio).
Uma classe funciona como uma "receita" para criar objetos. Inclusive, vários objetos podem ser criados a partir de uma única classe (assim como várias casas/prédios poderiam ser construídos a partir de uma única planta; ou vários bolos poderiam ser preparados a partir de uma única receita).

Representação gráfica das classes



Padronização dos nomes de uma classe

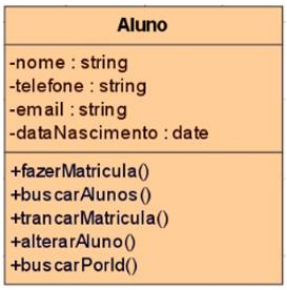


Nome da Classe: Minúscula com primeira letra maiúscula = Aluno, Professor e ContaReceber

Atributo: Minúsculo, se necessário primeira letra da segunda palavra maiúscula = nome, telefone e dataNasci.

Métodos: Minúsculo, se necessário primeira letra da segunda palavra maiúscula = inserir(), alterar() e buscarPorId()

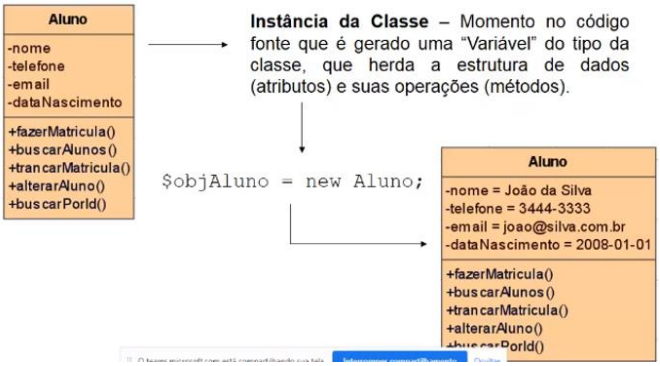
Tipos de Dados



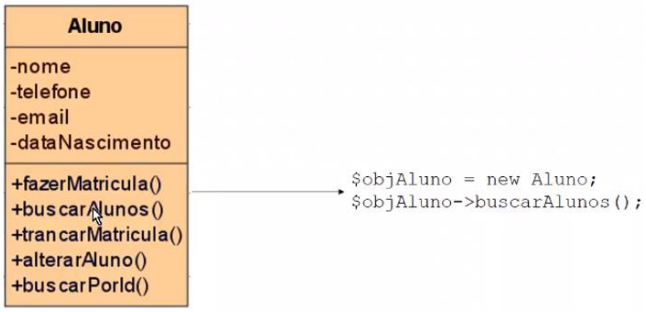
Tipos de dados:

- ✓ String
- ✓ Integer
- ✓ Boolean
- ✓ Date
- ✓ Double

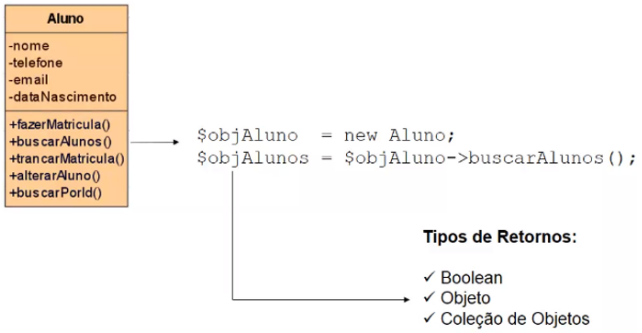
Criação do Objeto (programação)



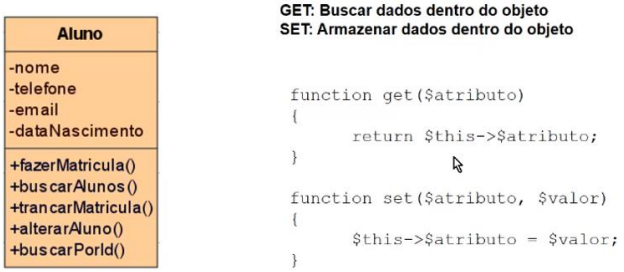
Chamada dos métodos



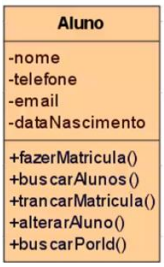
Retorno dos métodos



Métodos GET e SET



Referência aos atributos



Referência Fora da Classe

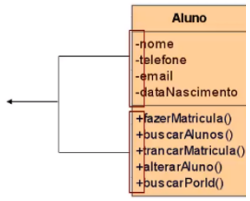
```
$objAluno = new Aluno;
print $objAluno->get('telefone');
```

Referência Dentro da Classe

```
print $this->telefone;
```

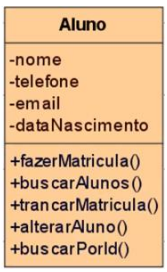
Visibilidade da classe, atributos e métodos

+ ou public → Acesso dentro ou fora da classes instanciada.
 # ou protected → Acesso dentro da classe ou pelas classes dependentes (herança).
- ou private → Acesso somente dentro da classe.
~ ou package → Acesso somente dentro do mesmo pacote.



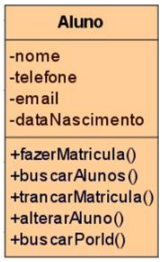
Visibilidade da classe

```
public class Aluno {
    // atributos
    // métodos
}
```



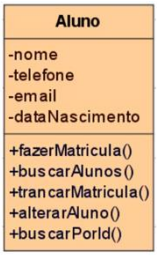
Visibilidade da classe e método

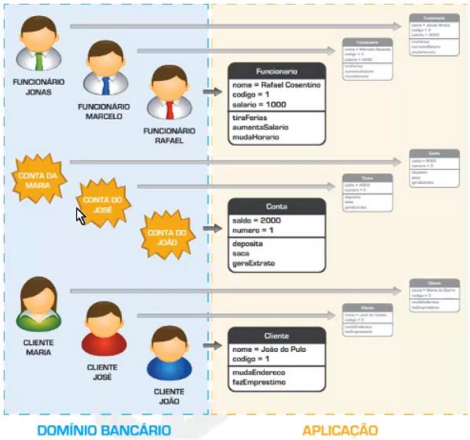
```
public class Aluno {
    // atributos
    public function buscarAlunos(){

    }
}
```



Visibilidade da classe, método e atributos

```
public class Aluno {
    private var $nome;
    private var $telefone;
    private var $email;
    private var $dataNascimento;
    public function buscarAlunos(){
    }
}
```





UML - Unified Modeling Language

Definição

É um conjunto de ferramentas (diagramas) que permitirão representar o modelo de um sistema qualquer, porém, por meio do paradigma de Orientação a Objetos.

Objetivos

A modelagem de sistemas (não apenas de software) usando os conceitos da orientação a objetos; Estabelecer uma união fazendo com que métodos conceituais sejam também executáveis; Criar uma linguagem de modelagem usável tanto pelo homem quanto pela máquina.

Histórico

Nas décadas de 1980 e 1990 houve uma explosão de metodologias para desenvolvimento software; Existência de diversas maneiras para representar um software; Entre 1994 e 1996, surge a versão 0.9 da UML e em 1997 é aprovada como padrão para desenvolvimento orientado a objetos.

O que é a UML?

A UML é uma linguagem padrão para visualizar, especificar, construir e documentar os artefatos de um sistema intensamente baseado em software.
A UML combina:
- Conceitos de Modelagem de Dados (Diagrama Entidade-Relacionamento);
- Modelagem de Negócios (Fluxo de trabalhos);
- Modelagem de Objetos;
- Modelagem de Componentes.

Pode ser usada com todos os processos, durante todo o ciclo de desenvolvimento e com diferentes tecnologias de implementação.

Modelagem Visual

Modelagem visual permite a construção de forma correta.
- Compreensão dos requisitos dos usuários.
- Validação do design.
- Visualizar interface, lógica de negócio e dados separadamente.
- Separar domínios de negócio conforme apropriado.
- Visualizar todas as dependências.
- Validar performance antes do código começar.

Usos da UML

Mostrar a interação da aplicação com o mundo interno e externo; Mostrar as interações dos usuários com o sistema; O comportamento da aplicação; A estrutura do sistema; Os componentes do sistema; A arquitetura da empresa.

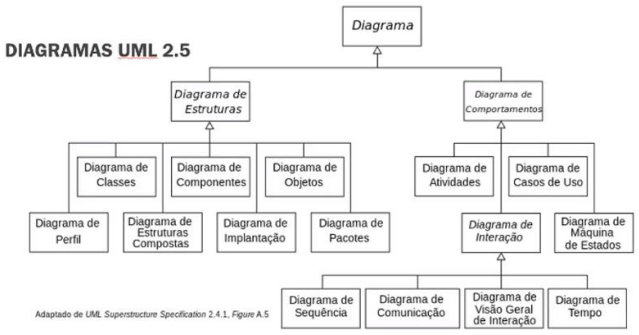


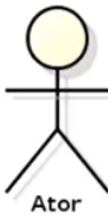
Diagrama Use Case (Casos de Uso)

É o diagrama mais abstrato, flexível e informal da UML. Normalmente, é utilizado no início da modelagem para identificar os requisitos do sistema; São utilizados para captar o comportamento pretendido do novo sistema (macro atividades); Pode ser utilizado como base para criação de outros diagramas; Estão associados aos requisitos funcionais do sistema; Representam o comportamento do sistema do ponto de vista do usuário; Usando uma linguagem simples, permite que qualquer pessoa compreenda o comportamento externo do sistema.

Componentes principais

O modelo de casos de uso tem como principais componentes:
- Atores = entidade externa que interage com os casos de uso;
- Casos de uso = especifica uma funcionalidade completa do sistema;
- Relacionamentos = mostra a comunicação entre o ator e o caso de uso.

Ator



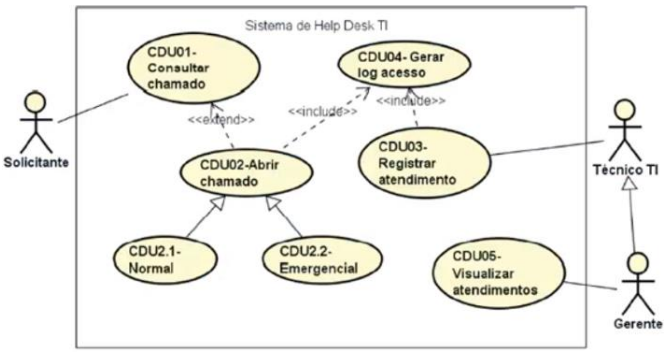
Representam os papéis desempenhados pelos diversos usuários que poderão utilizar ou interagir com os serviços e funções do sistema; Pode ser qualquer elemento externo que interaja com o sistema, inclusive um software ou hardware; Exemplos típicos: cliente, aluno, dispositivo de conexão de rede, sistema de faturamento etc.

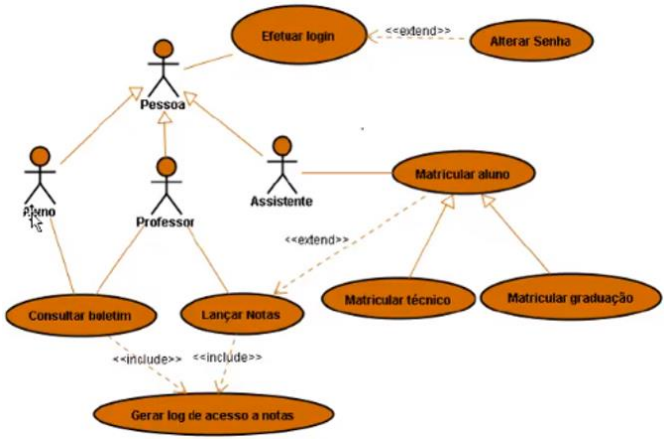
Identificando Atores:
Algumas perguntas podem auxiliar a identificar atores:
- Quem utiliza a principal funcionalidade do sistema?
- Quem vai precisar de suporte do sistema para realizar suas tarefas diárias?
- Quem precisa manter, administrar e deixar o sistema "rodando"?
- Quais dispositivos de hardware o sistema precisa manipular?
- Com quais outros sistemas o sistema precisa interagir?
- Quem ou o que tem interesse nos resultados produzidos pelo sistema ?

Casos de Uso

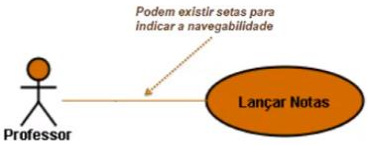
Casos de uso são utilizados para modelar os requisitos funcionais do sistema;
Descrevem as funcionalidades do sistema, conforme esperadas pelos usuários, retratando um "diálogo" que uma entidade externa realiza com o sistema; Um caso de uso é baseado em um cenário descritivo, mostrando como o ator interage com o sistema; Deve ser completo, isto é, não possível "quebrar" um caso de uso grande em diversos casos de usos menores; Realizado em nome de um ator que, por sua vez, deve pedir direta ou indiretamente ao sistema tal realização; Pode se relacionar com um ou mais atores através de associações.

Identificando Casos de Uso
- Quais funções o ator requer do sistema? O que o ator precisa fazer?
- O ator precisa criar, ler, destruir, modificar ou armazenar algum tipo de informação dentro do sistema?
- O ator precisa ser notificado de eventos do sistema? O ator precisa notificar o sistema sobre algum evento?
- O trabalho diário do ator poderia ser simplificado ou tornado mais eficiente através de novas funcionalidade do sistema?
- Quais entradas e saída o sistema precisa?
- Quais os principais problemas com o atual sistema?





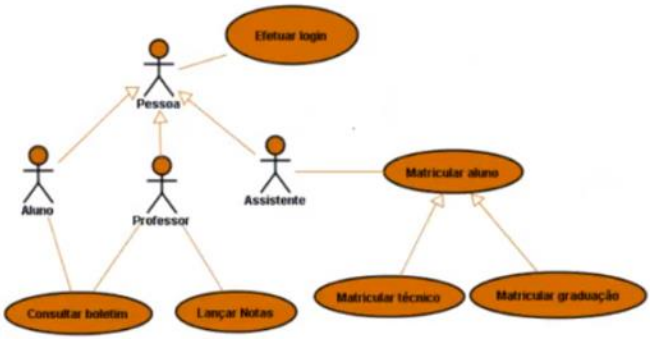
Relacionamentos - Associação



São representadas por uma linha que liga o ator ao caso de uso.

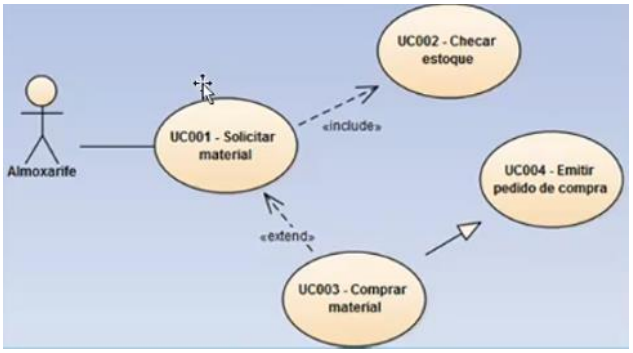
Generalização/Especialização

Forma de associação na qual existem dois ou mais casos de uso com características semelhantes; Existem pequenas diferenças entre os casos de uso associados; Também é possível com atores.



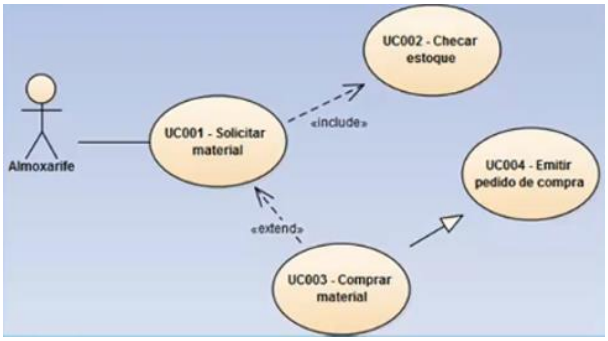
<<INCLUDE>>

A execução de um caso de uso obriga a execução de um outro; Pode ser comparado à chamada de uma sub-rotina.



<<EXTEND>>

Descreve cenários opcionais de um caso de uso. Só ocorrerá se uma determinada condição for satisfeita.



Exemplos

1 – Encomenda de placas

João confecciona placas por encomenda. Como o volume dos pedidos tem aumentado, ele pediu ao filho que fizesse uma pequena aplicação que controle:

- o cadastro de seus clientes;
- as encomendas;

Quando ele recebe uma encomenda, João anota num caderninho o nome do cliente e o seu telefone.

Para a encomenda, ele registra: o tamanho da placa (altura e largura), a frase a ser escrita, a cor da placa (branca ou cinza), cor da frase (azul, vermelho, amarelo, preto ou verde), data de entrega, valor do serviço e valor do sinal.

A aplicação deve obrigar que o valor do sinal seja, de no mínimo, 50%. Para calcular o valor da placa, as seguintes fórmulas são usadas:

Área = altura x largura

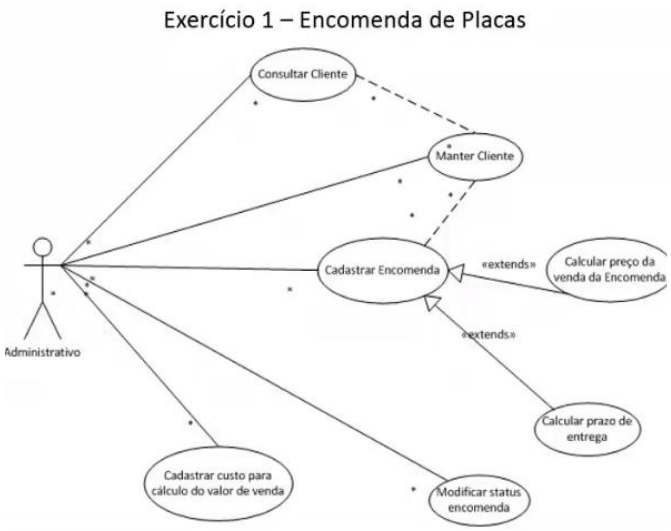
Custo_material = área x R$ 147,30

Custo_desenho = número_letras x R$ 0,32

Valor_placa = custo_material + custo_desenho

Para calcular o prazo de entrega, considera-se que ele só consegue produzir seis placas por dia.

João deseja que o sistema controle os pedidos, calcule o preço final das peças e o prazo de entrega. Para cada encomenda cadastrada, deve ser emitido um recibo em duas vias (cliente e empresa), contendo todos os dados da encomenda e do pagamento.

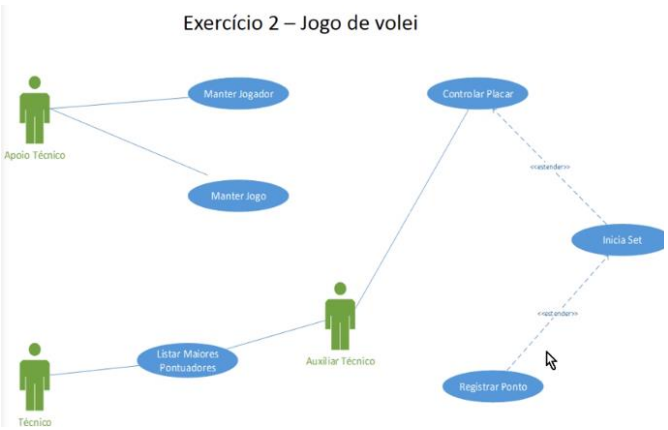


2 – Um treinador de vôlei deseja levar seu notebook para os jogos a fim de obter:

- controle do placar
- controle dos pontos de cada partida, identificando-os como: ponto de saque, ponto de ataque (quando a vantagem estiver com o time adversário), ponto de contra-ataque (quando a vantagem estiver com o próprio time), pontos de bloqueio, erro do adversário. No caso de bloqueio é necessário cadastrar se foi individual, duplo ou triplo.

São requisitos para a implantação dessa aplicação:

- cadastrar o nome de todos os jogadores do time e o número de suas camisas;
- para cada jogo agendado, cadastrar: a data e hora do jogo, o local, o nome do time adversário, os nomes dos juízes (principal e auxiliar);
- a aplicação deve exibir para controle em cada set o placar que pode ser alterado pelo auxiliar técnico, informando quem fez o último ponto e o tipo de ponto. No caso de o ponto ser do time adversário, basta identificar o tipo de ponto;
- ao final de um jogo, o sistema deve exibir a lista dos maiores pontuadores e o somatório de pontos, por tipo, do jogo.



3 - A empresa Yep implantou uma senha de atendimento para o SAC de suas lojas. O objetivo é reduzir o tempo de espera na fila.

O atendimento é dividido por assuntos e cada caixa pode cuidar de um ou mais assuntos, ou um assunto pode ser tratado por um ou mais caixas. Para cada caixa, deve-se saber o número e a posição (direita ou esquerda da máquina de senhas).

Para cada caixa, deve-se ter um histórico de atendimentos para se obter estatística.

A estatística deve ser detalhada quanto ao tempo mínimo, médio e máximo de atendimento por caixa e por dia, além do número de atendimentos por assunto.

A qualquer momento é preciso saber que caixa está com um determinado número de atendimento.

O administrativo controla os assuntos, os caixas, obtêm estatísticas e relatórios. O caixa é o responsável por controlar a próxima senha e o balcão de informações gera as novas senhas.