**Anotações e Estudo T.I**

O que é e como funciona:

* **ServiceNow:** Companhia norte-americana de computação em nuvem que fornece softwares de gerenciamento de serviços como um serviço por meio de uma plataforma de nuvem corporativa de serviços de dados único. Especializada em gerenciamento de serviços de TI (ITSM), gerenciamento de operações de TI (ITOM) e gerenciamento de negócios de TI (ITBM).
* **SalesForce:** É uma empresa de softwares que foca na solução de gerenciamento de relacionamento para aproximar empresas e pessoas. É uma plataforma de CRM integrada que oferece a todos os departamentos uma visão única e compartilhada de cada cliente.
* **Cloud:** Computação em nuvem é o conceito que se aplica ao acesso de um conteúdo que está num ambiente externo, que não nos pertence, por meio da internet. Entrega vários tipos de serviços, não somente em T.I.
* **Java:** Linguagem de programação orientada a objetos super versátil.

**Introdução a Lógica de Programação**

Muito relacionado a computação e á sua história. Sempre houve lógica, ela sempre esteve presente, mas apenas conceituada e formalizada por Aristóteles na Grécia antiga. Se baseia nos princípios racionais e nas argumentações válidas. Existem vários tipos de lógica: Matemática aristotélica, simbólica, etc. Apenas nos concentraremos na algébrica e formal.

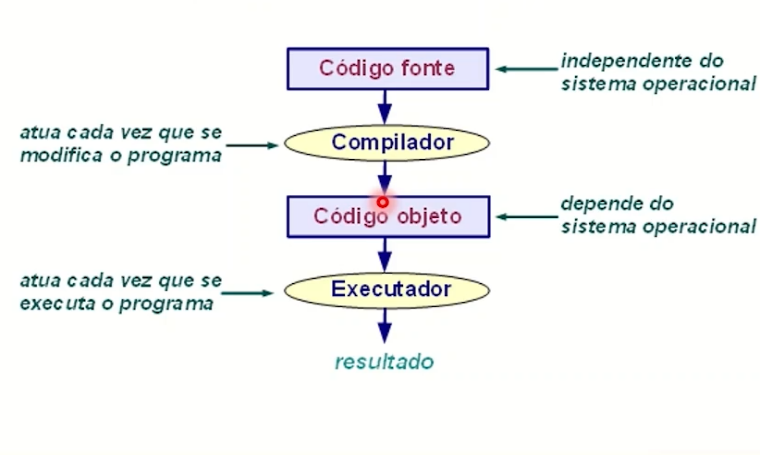
Para existir lógica não precisa haver fatos verdadeiros, apenas um ponto de início e um ponto final que faça sentido, ou seja, a lógica consiste na construção racional e coerente do raciocínio.

**Lógica Algébrica**

Veio à tona a partir dos trabalhos de George Boole e Augustus de Morgan e possibilitou o aprofundamento na programação. Sendo assim a Álgebra Booleana está relacionada tanto na construção de softwares quanto de Hardwares

**Linguagens de Programação Compiladas e Interpretadas**

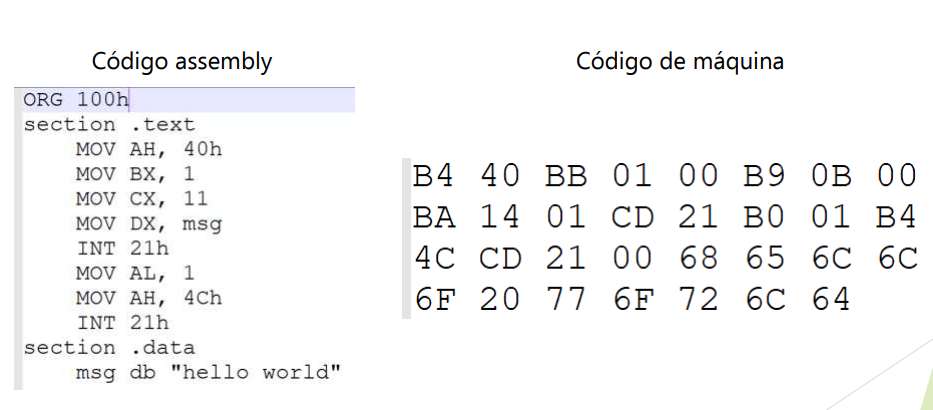
Nas compiladas utilizamos um programa de compilação para traduzir um código fonte de uma linguagem para outra para que o sistema operacional a interprete.



**Assembly:** É o nome dado ao código compilado e simples, se torna outra linguagem. Nela sabemos exatamente o que o processador está fazendo. Não é muito fácil de entender, mas é exatamente o que roda no cérebro do pc.

**Assembler:** É o nosso executador, que lê o arquivo contendo os comandos simples do assembly e os transforma em linguagem de máquina.

**Linguagem de máquina:** É a única coisa que o processador consegue interpretar, são apenas códigos, não dá para ser lido por humanos (Normais).



Nas interpretadas apenas utilizamos um interpretador como um navegador para rodar o conteúdo por exemplo.

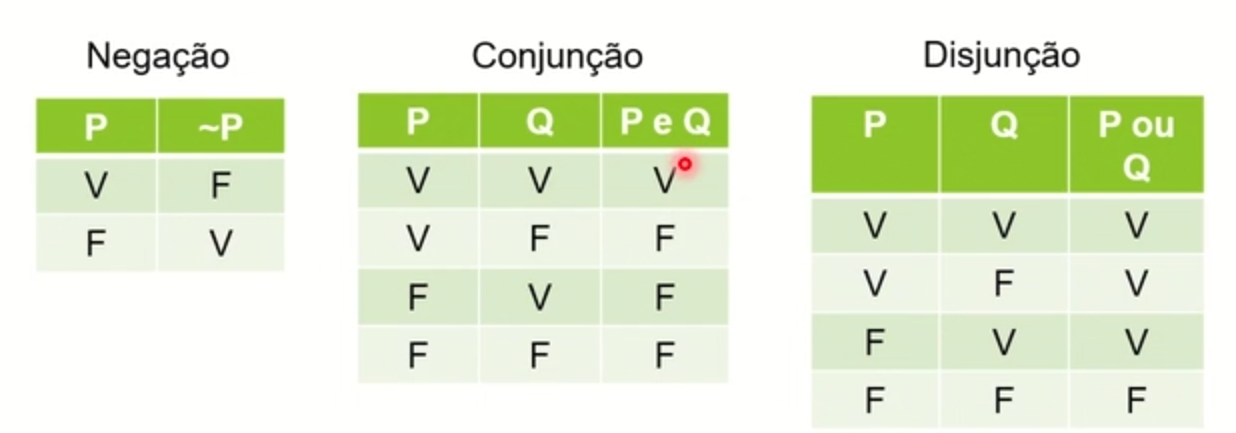
Resumo: Nas compiladas a tradução é feita apenas uma vez. Já nas interpretadas a tradução é feita sempre quando se executa o software com o conteúdo.

**Lógica na Solução de Problemas**

**Algoritmo:** Sequência de ações que obedecem a uma ordem que possuem de um estado inicial durando determinado tempo resultando em estados esperados e bem definidos. Fazemos isso até inconscientemente no dia a dia, como comer, respirar ou tomar banho.

**Lógica Formal:** Ferramenta que nos ajuda a desenvolver uma cadeia de pensamentos reconhecendo possíveis falhas, identifica raciocínios válidos e não válidos.

**Álgebra Booleana:** Entrando na programação a importância da álgebra booleana está na sua utilização de símbolos e operações algébricas para representar proposições e suas inter-relações.



Algoritmos

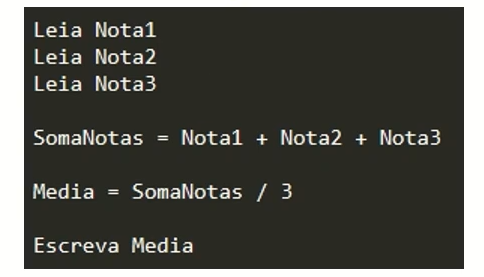
Conjunto de ações perfeitamente definidos com números finitos de etapas antecipando falhas para concluir seu objetivo. DEVE SER CLARO E OBJETIVO.

Estes são os passos que devemos seguir para aplica-los na programação:

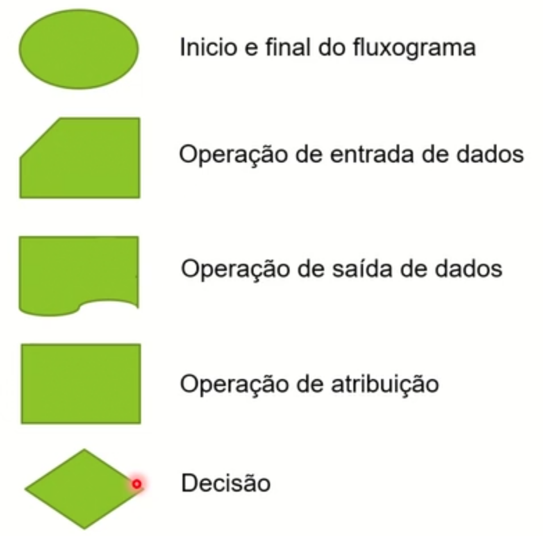


pseudocódigo: Descrição como algoritmo.

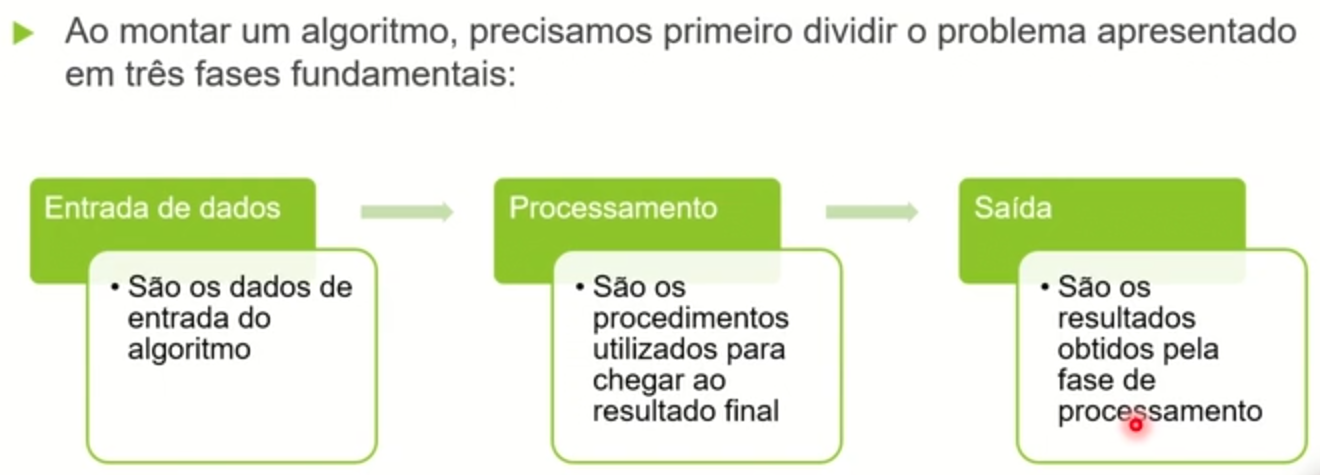
Ex - Somar as 3 notas de 3 provas diferentes de um aluno a fim de conseguir uma média.



Fluxograma: Representação gráfica como algoritmo.



Construção do Algoritmo:



**Módulo 1**

**Operadores**

São os símbolos ou palavras que utilizamos na lógica de programação.

Sendo eles:

* **Literal ou String:** Só pode ser somado, aceita vários caracteres;
* **Carácter ou Char:** Só pode ser somado, aceita apenas um caractere;
* **Inteiro:** Pode ser somado, subtraído e multiplicado;
* **Real:** Pode ser divido;
* **Lógico:** Diz ser verdadeiro ou falso.
* **^:** Potência.
* **Escolha:** Para determinar alternativas simples.
* **Fimescolha:** Finaliza a escolha.
* **Caso:** Para atribuir uma situação na escolha.
* **Casonao:** Para atribuir nova situação caso a primeira não funcione (Bem parecido com se e senao).
* **Div:** Apresenta somente o número inteiro da divisão.
* **Mod ou %:** Apresenta o restante da divisão.
* **Enquanto ou WHILE:** Calcula ou executa determinada situação automaticamente em loop, se não houver uma saída o while continuará infinitamente.
* **Fimenquanto:** Finaliza o enquanto.
* **Repita:** Só funciona quando se pode enxergar um fim e repetirá a sentença até ter a resposta desejada como saída.
* **Ate:** Finaliza o repita (Não conta como flag de saída!).
* **Para:** Este operador pode substituir o enquanto e o repita em algumas situações, deixando o código mais limpo visualmente, muito usado para vetores e banco de dados.
* **Passo:** Ao atribuir a tag "passo" no operador “para”, você determina uma regra de multiplicação, de em quantos números serão exibidos.
* **Aleatório:** Sorteia um valor aleatório na execução
* **Comparadores de operação:**
* **Se:** Compara;
* **Então:** Atribui;
* **Senao:** Substitui o se caso não funcione;
* **Fimse:** Finaliza a comparação.
* **Tabela Verdade:** Podem diminuir as comparações!
* **E:** Apenas válido quando toda a afirmação é verdadeira.
* **Ou:** Válido mesmo quando uma parte da afirmação é falsa, mas a outra verdadeira.

**Módulo 2**

**Procedimentos, Banco de Dados e Orientação de Objetos**

**Debug:** É analisar um código linha por linha, facilita para encontrar um erro. Cada linguagem de programação tem uma forma de fazer.

**Vetores (Array):** É como um baú capaz de armazenar e executar em apenas uma variável várias outras, quantas quiser ou precisar.

**Matrizes: Tipos de vetores de alta dimensão.**

**Procedimentos:** Método para organizar variáveis. Processa de acordo com dados de entrada.

**Função:** Processa com resposta em retorno.

**Banco de dados**

**SQL:** Stuctured Query Language, ou Linguagem de Consulta Estruturada, é a linguagem de pesquisa declarativo padrão para banco de dados relacional. Muitas das características originais do SQL foram inspiradas na álgebra relacional.

**XAMPP:** É um pacote com os principais servidores de código aberto do mercado, incluindo FTP, banco de dados MySQL e Apache com suporte às linguagens PHP e Perl.

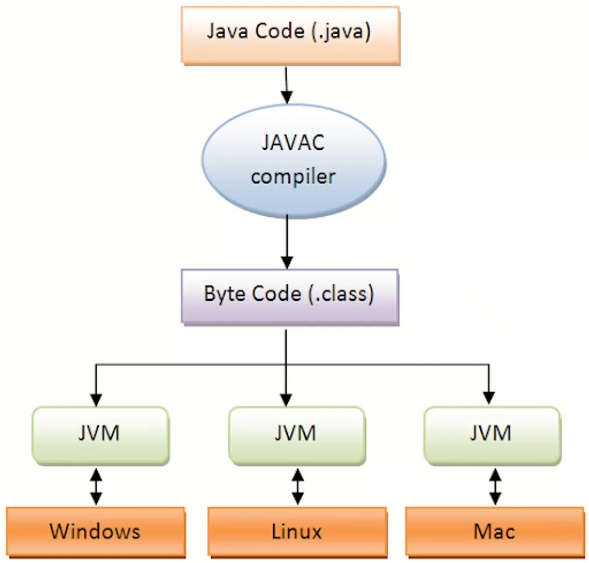
**Primary Key:** Transforma um item em algo único no sistema e fácil de achar pelo seu ID.

**Inner Join:** Permite a junção entre duas ou mais tabelas, desde que tenham a ver uma com a outra.

**Java**

Criado em 1995 por uma equipe chefiada por James Gosling na empresa SunMicrosystem

de fácil compilação e executável pela maioria dos sistemas operacionais.



|  |
| --- |
| **Relacionais:**   * >: * <: * >=: * <=: * ==: * !=: |

Operadores em JAVA

**Primitivos:**

|  |
| --- |
| **Atribuição:**   * =: * +=: * -=: * \*=: * /=: * %=: |

|  |
| --- |
| **Aritméticos:**   * +: Adição; * -: Subtração; * \*: Multiplicação; * /: Divisão; * %: Porcentagem. |

|  |
| --- |
| **Atribuição:**   * =: * +=: * -=: * \*=: * /=: * %=: |

|  |
| --- |
| **Lógicos:**   * &&: * ||: * !: |

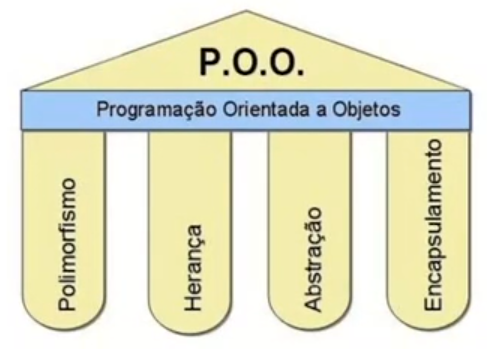
* Bitwise:
* &:
* |:
* ^:
* ~:
* <<:
* >>:
* >>>:
* Condicional (ternário): ? : :
* **Não Primitivos:**
* Instanceof: Verifica se um objeto é uma instancia de uma determinada classe ou interface, se um objeto é do tipo especificado. Ex:
* Ternário(? :): Permite executar uma determinada ação baseada em uma condição. É uma forma compacta de escrever uma estrutura condicional “if-else”. Ex:
* Cast ( (Tipo) ):

Classes em JAVA

* **Primitivas:** São simples e se limitam a valores em bits. Em Java há 8 delas:
* Int: Pode ser somado subtraído e multiplicado, sendo o mais recomendado para trabalhar com números inteiros;
* Float: Armazena valores numéricos em ponto flutuante (que representa números com casas decimais, tanto negativos quanto positivos) de precisão simples;
* Double: Armazena números com casas decimais;
* Char: Armazena apenas um caractere, nele utiliza-se aspas simples (‘’);
* Byte: Parecido com o int;
* Boolean: Armazena valores lógicos (True ou False);
* Long: Representa números inteiros de 64 bits, com tamanho de memória de 8 bytes, geralmente é usado para trabalhar com valores muito grandes;
* Short: Representa números inteiros de 16 bits, com tamanho de memória de 2 bytes, geralmente usado quando é necessário economizar espaço e memória.
* **Operadores Lógicos:**
* &&: Apenas válido quando toda a afirmação é verdadeira;
* ||: Válido mesmo quando uma parte da afirmação é falsa, mas a outra verdadeira.
* !: Negação
* **Operadores Relacionais:**
* >: Maior;
* >=: Maior ou igual;
* <: Menor;
* <=: Menor ou igual;
* ==: Compara igualdade;
* !=: Diferente.
* If/else: Avalia expressões como verdadeira ou falsa de acordo com uma situação predeterminada pelo usuário, executando uma ou outra rotina;
* Switch: Para definir escolhas;
* Case: Para definir um resultado para determinada escolha;
* Break: Para finalizar o Case.
* Default: Será executado em último caso quando nenhum dos Case for.
* **Estruturas de controle de fluxo:**
* For: Utiliza-se quando se sabe até onde quer chegar.
* While: Repete uma instrução até que uma expressão booleana seja verdadeira.
* True: O while ficará em loop infinito.

**Orientação a Objeto**

Veio do SmallTalk em 1960. É uma forma de resolver problemas de uma maneira não mais estruturada pensando no problema e seus agentes como real, trazendo isso para o algoritmo. Esses são os seus 4 princípios:

****

**Polimorfismo:** Princípio no qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que tem a mesma identificação (assinatura) mas comportamentos distintos, especializados par cada classe derivada, usando para tanto uma referência a um objeto do tipo de superclasse. Resumidamente é como uma palavra sinônima de outra, que pode se aplicar a mesma situação das palavras sinônimas e a outras aplicações distintas.

**Herança:** Permite criar uma nova classe a partir de uma já existente. Também chamadas de subclasses.

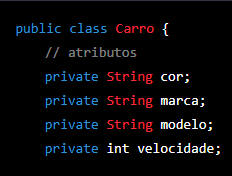
**Abstração:** Reconhecer uma entidade real e passa-la para um algoritmo, extraindo suas informações necessárias para resolver um problema.

**Encapsulamento:** Vinculado ao conceito de segurança, o encapsulamento tem a ver com proteger uma informação ou esconde-la.

**Conceitos básicos de P.O.O:**

**Encapsulamento:** Garante a segurança das informações do objeto

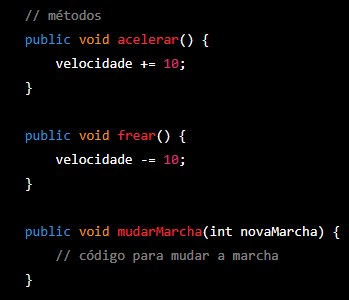
**Classe:** Um modelo para criar objetos e determinar um grupo com suas variáveis comuns. Ex: Carro, que hospeda vários modelos, marcas, cores, etc. Ou Pessoa, que armazena nome, idade, altura, etc; Ex:



**Objeto:** Entidade abstrata ou concreta que agrupa um conjunto de informações. Ex: Echy possuí nome, idade, gênero e pertence à classe pessoa.

**Atributo:** Elementos que definem estrutura de uma classe, são chamados também de variáveis de classe e podem ser divididos em dois tipos básicos: atributos de instância e de classe. Os valores dos atributos de instância determinam o estado de cada objeto, enquanto os da classe determinam o estado da classe, se ela é privada ou pública por exemplo. Os métodos da classe mesmo se ela for privada terão acesso a ela (Afinal são métodos que estão dentro da classe...)

**Método:** Procedimento que atribui responsabilidades ao objeto, o que ela vai executar no algoritmo caso instanciada. Para que seja possível chamar apenas o método é necessário colocar “static” entre public e void. É possível ter métodos com os mesmos nomes desde que sua assinatura seja diferente;



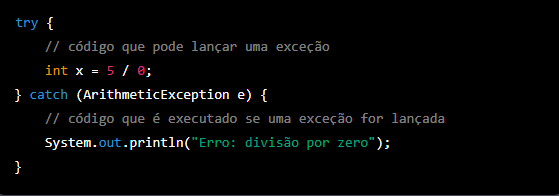
**Modificador de acesso:** Também chamado de visão de método ou visão de atributo, é a palavra-chave que define um atributo, método ou classe como público, privado ou protegido. Public o deixa aberto a modificações externas do usuário de forma mais simples, enquanto o Private o deixa privado e passa a ser necessário um método de acesso para poder modificar o valor da variável dentro da class;

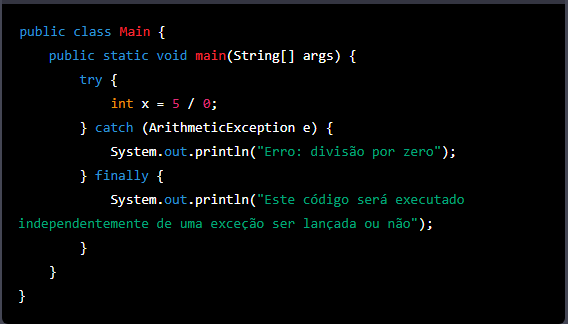
**Método de acesso:** São os chamados getters e setters. Dão acesso a entidades externas para o usuário acessar informações do objeto. Dentro deles podemos fazer validações especificas!

**Método Construtor:** Sempre nomeado com mesmo nome da classe, sendo sempre chamado quando a própria classe é instanciada.

**Herança:** Já mencionada acima, permite criar uma nova classe a partir de uma já existente. Também chamadas de subclasses. Pode ser identificada quando percebemos que “há um tipo de” alguma coisa. Ex: Vendedor é um tipo de funcionário; Então: Vendedor extends Funcionário;

Static: Indica que um método pode ser chamado sem precisar instanciar uma classe;

Try, Catch e Finally: Geralmente usados para tratamento de código. O Try pode lançar uma exceção (Exception), no qual executará o catch caso essa exceção aconteça. Ex:

Enquanto ao Finally, ele pode ser usado para executar o try independente se há exceções ou não. Ex:

**Passo a passo para criar um objeto funcional:**

1 – Crie o arquivo tipo Class que hospedará o objeto com seu nome;

2 – Defina seus atributos globais;

3 – Defina quais serão seus tipos de métodos (Acessores, construtores, etc)

4 – Quando terminar, basta instanciá-lo (chama-lo) no código principal;

Construção de Interfaces Gráficas em Java

**Conceitos básicos**

**Criando uma tela**

**Paleta de componentes**

**Componentes**

**Propriedades**

**Eventos**

**Implementando eventos**

**Arvores de componentes**

7b 1h e 10 min

Pulei, veja depois;

Projeto Final do Curso

**Definição básica do projeto**

- Tema: To do App;

- Descrição: Aplicação para gerenciamento de projetos e tarefas;

- Objetivo: Organizar projetos;

- Entidades:

* Projeto:

1. Nome;
2. Descrição;
3. Data de Criação;
4. Data de Atualização.

* Tarefa;

1. Nome;
2. Descrição;
3. Status;
4. Observações;
5. Prazo;
6. Data de Criação;
7. Data de Atualização.

Requisitos

- Permitir cadastro de projetos e de tarefas;

- Permitir criar, deletar e alterar projetos e tarefas;

Regras

- Sem sistema de login;

- Sem conceito de usuário;

Fazer depois:

- Deve ser possível filtrar as tarefas por tag;

- Toda tarefa deve pertencer a uma lista;

- Não podem haver tags repetidas;

**Padrão utilizado:** MVC (Model View Controller).

Nesse padrão há classes que representam as entidades do sistema. Em Model há as entidades em si (como users), em View há as classes da interface e em Controller há as que controlam tudo em si, é a aplicação de tudo;

**SGBD** (Sistema de gerenciamento de banco de dados): MySQl;

**Conectando ao MySQL**

Para conectar ao banco de dados comece criando uma variável do tipo DRIVER que permita isso e aplique o JDBC (Java Database Connectivity). Ex: public static final String DRIVER = “com.mysql.jdbc.Driver”.

Para ativar o JDBC como dependência do Java é necessário informar isso ao Gradle (Alteramos isso no arquivo build.gradle em “dependencies”). O código pode ser achado no seguinte site para MySQL: //<https://mvnrepository.com/artifact/mysql/mysql-connector-java>;

Statement: Objeto que ajuda a preparar comando sql para ser executado no banco de dados, usando “.set” + o tipo da variável é possível alterá-la;

ResultSet: Objeto que guarda valores. Para chama-lo em um statement utilizamos o ExecuteQuery;

11 L

**Java Script**

Java Script e Java são duas linguagens diferentes, a razão pelos nomes serem parecidos é que o JS se inspirou no nome Java para repercutir e ganhar fama, foi uma jogada de marketing. Foi criada em 1995.

**Estrutura JS no HTML:**

<!DOCTYPE html>

<html lang="pt-br">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Estrutura</title>

    <script>

        //O Código JS fica aqui!!

    </script>

</head>

<body>

</body>

</html>

**Pop-up:** Alert gera um pop-up ou aviso.

<script>

   //Exibindo pop-up:

   window.alert("Yey");

</script>

**Variáveis**: É um espaço reservado de memória que armazena um valor.

<script>

        //Variaveis

        var nome = "Echy"; //string

        var idade = 19; //Int

        var altura = 1.65; //float

        var estuda = true; //boolean

        //exibindo

        window.alert("Nome: " + nome);

        window.alert("Idade: " + idade);

        window.alert("Altura: " + altura);

        window.alert("Estuda: " + estuda);

    </script>

**Operadores relacionais:** São os sinais mais comuns para fazer operações.

<script>

        //maior

        window.alert(5 > 4);

        //menor

        window.alert(5 < 4);

        //igual

        window.alert(5 == 4);

        //maior ou igual

        window.alert(5 >= 4);

        //menor ou igual

        window.alert(5 <= 4);

        //diferente

        window.alert(5 != 4);

</script>

**Condicional simples (if, else):** Quando se tem apenas uma resposta simples.

<script>

        //variavel

        var idade = 19;

        //condicional

        if(idade >= 18){

            alert("Aprovado!");

        } else {

            alert("Reprovado :(")

        }

    </script>

**Condicional aninhada (if, else if...):** Quando se tem diversas respostas.

<script>

        //media

        var media =8.5;

        //condicional

        if(media == 10){

            alert("Kudos!")

        } else if (media >= 9){

            alert("Nice!")

        } else if (media >= 8){

            alert("Good.")

        } else if (media >= 7){

            alert("On media.")

        } else if (media >= 5){

            alert("On exam.")

        } else {

            alert("disapproved :(")

        }

    </script>

**Operador lógico E (&&):** Compara duas ou mais informações, mas só aceita se todas forem verdadeiras.

<script>

        //variáveis

        var media = 10;

        var faltas = 20;

        //condicional

        if ((media >= 7) && (faltas <= 15)){

            alert("Aprovada!");

        } else {

            alert("Reprovada :(");

        }

    </script>

**Operador lógico Ou (||):** Compara duas ou mais informações e aceita tudo mesmo que uma seja falsa.

<script>

        //variáveis

        vsar trab1 = 8;

        var trab2 = 5;

        //condicional

        if ((trab1 >= 7) || (trab2 >= 7)){

            alert ("Aprovado!")

        } else {

            alert ("Reprovado...")

        }

    </script>

**Operador lógico não (!):** Negativa informações booleanas.

<script>

        //variáveis

        var trab = true

        //condicional

        if (!trab == true){

            alert("Você trabaia")

        } else {

            alert ("Você não trabaia")

        }

    </script>

**Estrutura de escolha (switch, case):** Nos ajuda a escolher entre opções!

<script>

        //variavel

        var codigo = 3;

        //Estrutura de escolha

        switch (codigo) {

            case 1:

                alert("Pizza");

                break

            case 2:

                alert("Coxinha");

                break

            case 3:

                alert("Hambúger");

                break

            default:

                alert("código invalido...");

       }

    </script>

**Laço enquanto (while):** Cria loops!

<script>

        //variavel

        var i = 1;

        //Laço de repetição

        while(i >= 3){

            alert("Oi :D" + i);

            i++;

        }

    </script>

Laço faça enquanto (do, while): Primeiro força uma ação e depois faz uma validação.

<script>

        //variavel

        var i = 1;

        //Laço

        do{

            alert(i);

            i++;

        } while (i <= 3);

    </script>

Laço para (for): Trabalha com 3 parâmetros, sendo eles o valor inicial, o valor final e sua incrementação.

<script>

        //laço de repetição

        for (var i = 1; i <= 3; i++) {

            alert("O indice vale: " + i);

        }

    </script>

Array(vetores): Banco de dados temporário que pode armazenar uma lista de coisas.

<script>

        //vetor

        var cursos = ["HTMl","CSS","JS"]; //1, 2, 3...

        //exibir

        alert("cursos[2]")

    </script>

Vetor de Objetos: Agrupa dentro da lista outras características do objeto.

<script>

        //vetor

        var cursos = [

            {nome:"HTML", carga: 20, sala: 3},

            {nome:"CSS", carga: 30, sala: 1},

            {nome:"JS", carga: 35, sala: 2}

        ]

        //exibir vetor de objetos

        alert(cursos[0].nome +" "+cursos[0].carga)

    </script>

**Funções:** São ações e interações (sempre usamos o “ponto” para chamar uma função). Existem várias funções prontas em JS, uma bem usada é a:

* .length: Conta caracteres.

<script>

        //Bastante usado para validar quantidade mínima de caracteres.

        //variável

        var termo = "Aprendendo JS";

        //Exibir quantidade de char

        alert(termo.length);

    </script>

* .toUpperCase e .toLowerCase: Deixa strings maiúsculas ou minúsculas.

<script>

        //variavel

        var palavra = "inFORmÁtica";

        //Maíuscula

        alert(palavra.toUpperCase());

        //Minuscula

        alert(palavra.toLowerCase());

    </script>

* .indexOf e .lastIndexOf: Localiza algo em especifico nesse caso é um caractere. O .lastIndexOf localiza o último caractere.

<script>

        //variavel

        var nome = "echyley";

        //Retornar a primeira letra E

        alert(nome.indexOf("e"));

        //Retornar ultima letra E

        alert(nome.lastIndexOf("e"));

    </script>

* .slice: Extrai ou corta elementos.

<script>

        //variavel

        var termo = "Aprendendo Java Script";

        //Extrair

        //alert(termo.slice(10));

        alert(termo.slice(10, 15));

    </script>

* .round: Função da biblioteca Math (Nos ajuda a usar matemática em JS) que arredonda valores.

<script>

//Math é uma biblioteca para trabalhar com matemática em JS

        //variavel

        var nota = "7.89";

        //Exibir nota formatada

        alert(Math.round(nota));

    </script>

* .max e .min: Também funções da biblioteca Math, max pra obter o maior valor e min para obter o mínimo.

<script>

        //obter menor valor

        alert(Math.min(-10, 5, 0, 100, -25, 2));

        //obter maior valor

        alert(Math.max(5, 7, 9, -13, 15));

    </script>

* .random: Da biblioteca Math, .random gera um número aleatório.

<script>

//Para se obter algum número desejado basta multiplicar por um valor em que esse número esteja incluído.

        //exibir número aleatório

        var numero = Math.random() \* 100;

        //Usamos o .round para arredondar.

        alert(Math.round(numero));

    </script>

* CRIANDO NOVAS FUNÇÕES:

<script>

//Você pode reutilizar funções e criar quantas quiser!

//Depois do nome da função em parenteses temos os parametros, que são as peças para a função trabalhar.

        //CRIANDO UMA FUNÇÃO

        //somar dois valores

        function somar*(num1, num2)*{

        //return exibe a função quando ela for chamada.

            return num1 + num2;

        }

        //CHAMANDO A FUNÇÃO

        alert(somar(5, 6));

//ao atribuir o valor 5 e 6, damos as peças para a função "somar" trabalhar.

    </script>

3 A