

Crie Um Sistema

Do portugol ao Spring Boot

“Aquilo que focamos sempre se expande, mas aquilo que você não dá atenção encolhe” Jonh Maxuel

Introdução

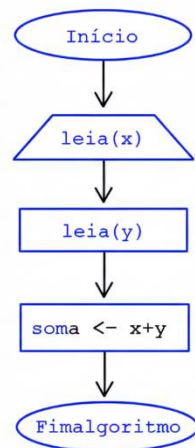
Uma variável é um espaço reservado na memória do computador. Essa variável tem um tipo que pode ser inteiro, real ou um caractere. Mas não basta ter variáveis, o computador precisa realizar cálculos aritméticos. Esses cálculos aritméticos processados pela ULA (Unidade Lógica e Aritmética) dentro do processador trabalham junto com os registradores tornando o acesso mais rápido guardando os valores dessas operações. Muito importante esse entendimento e aplicação em diversas linguagens de programação, mas para iniciantes utiliza-se do portugol para aprender os primeiros passos.

Assista esse vídeo que elaborei para visualizar na prática o uso do portugol criando variáveis.



Paradigma

Quando programamos devemos escolher um paradigma. Um paradigma é um modelo, exemplo a ser seguido. Um paradigma geralmente tem um conjunto de regras que moldam nossa forma de pensar. Quando pensamos de forma estruturada e sequencial seguimos um caminho pelo fluxograma da mente e pensamos top down. As tecnologias, porém, em sua evolução traz novos recursos, mas sempre de forma híbrida resgatam a ideia do estruturado e sequencial. Quando pensamos de forma sequencial e estruturada, organizamos nossa lógica de programação em fluxogramas.



Esse paradigma é chamado de paradigma estruturado. O nome estruturado vem do uso intensivo de estruturas:

- Estruturas de Condição
 - Se
 - Se Então Senão
 - Escolha
- Estruturas de Repetição
 - Faça
 - Enquanto
 - Repita Até
- Estrutura de Dados
 - Registros
 - Vetores
 - Matrizes
- Modularização
 - Funções
 - Procedimentos

A linguagem **Java, Python, C# , PHP , GO** entre outras utilizam o paradigma estrutura de forma híbrida, pois o foco dessas linguagens é a orientação a objetos.

“Muitos aqui nessa sala não seremos os mesmos” Jonh Maxuel

Para implementarmos esse mesmo algoritmo em uma **API Web Java RESTFull** é necessário primeiramente realizarmos uma ambientação.

Acesse o vídeo para aprender sobre a ambientação e a inicialização do nosso projeto:

- Spring Initializr
- Spring x Spring Boot
- Estrutura do Projeto
- Soma Dois Números
- Modelagem de Sistemas (UML)
- Mapeando as requisições
- Enviando dados

Execute a ação do vídeo e marque o tempo gasto:

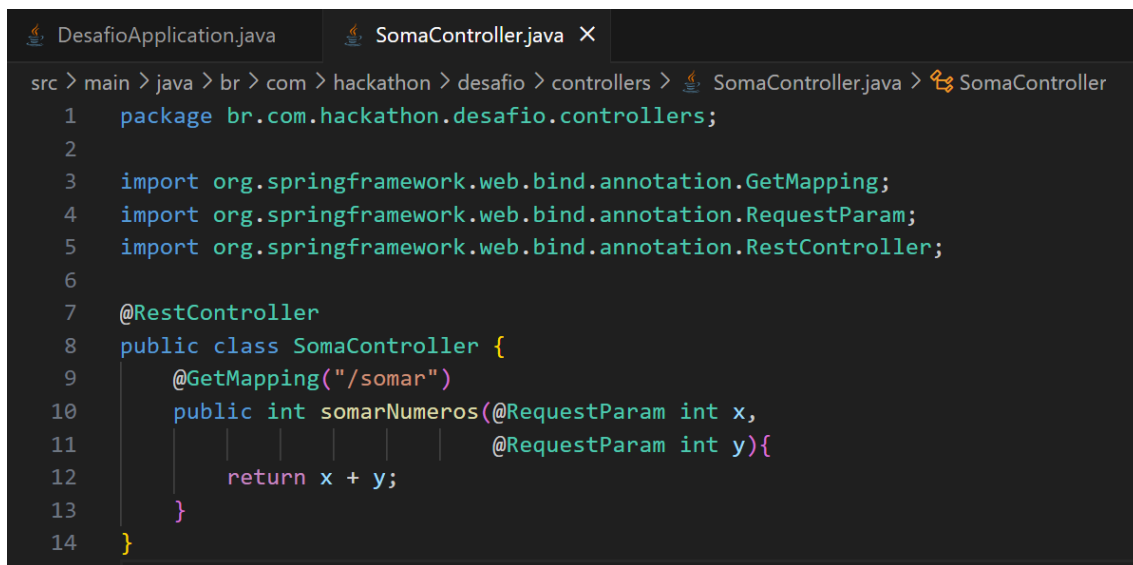
Competência	Tempo
Ambientar a API e criar um endpoint simples de somar dois números	

Para testarmos a **API** temos que realizar o download da **Insomnia**.

Como realizar o download e iniciar a sua utilização assista o vídeo:



Vamos agora criar o nosso Controller **SomaController** e as variáveis serão passadas por parâmetros via método **GET** do **HTTP**.



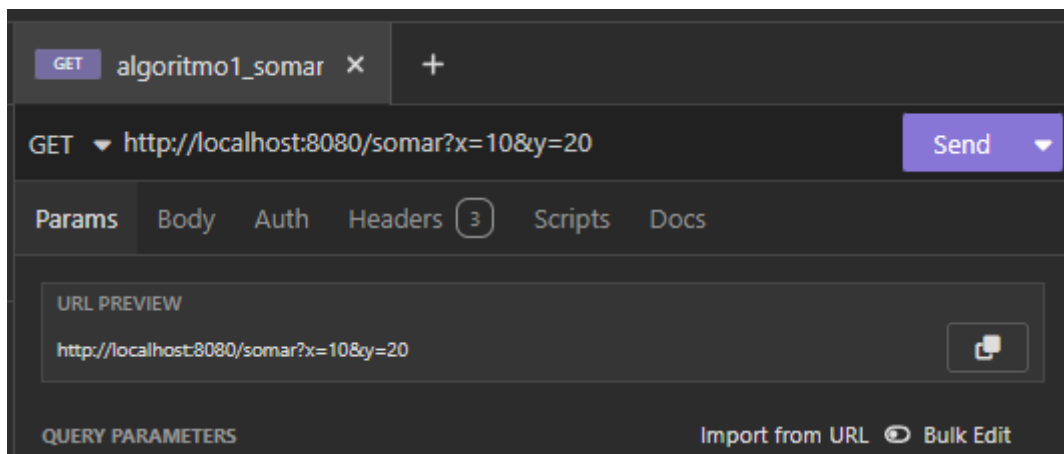
```

src > main > java > br > com > hackathon > desafio > controllers > SomaController.java > SomaController
1  package br.com.hackathon.desafio.controllers;
2
3  import org.springframework.web.bind.annotation.GetMapping;
4  import org.springframework.web.bind.annotation.RequestParam;
5  import org.springframework.web.bind.annotation.RestController;
6
7  @RestController
8  public class SomaController {
9      @GetMapping("/somar")
10     public int somarNumeros(@RequestParam int x,
11                             @RequestParam int y){
12         return x + y;
13     }
14 }

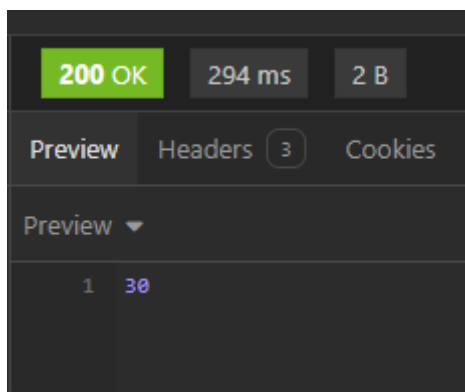
```

Figura 1 - implementação do algoritmo

Teste no Insomnia



Retorno:



Algumas considerações são importantes nesse desafio prático:

- **HTTP** é um protocolo web para comunicação entre cliente e servidor

- O Spring trabalha na visão do MVC (Model, View , Controller)
- O Controller é responsável pelas requisições e respostas e utiliza o padrão **HTTP** em sua estratégia de annotations.
- A importação de recursos para annotations é de suma importância para produtividade dentro do framework.
- O teste no **Insomnia** utilizou o método POST e as variáveis foram passadas pela URL.

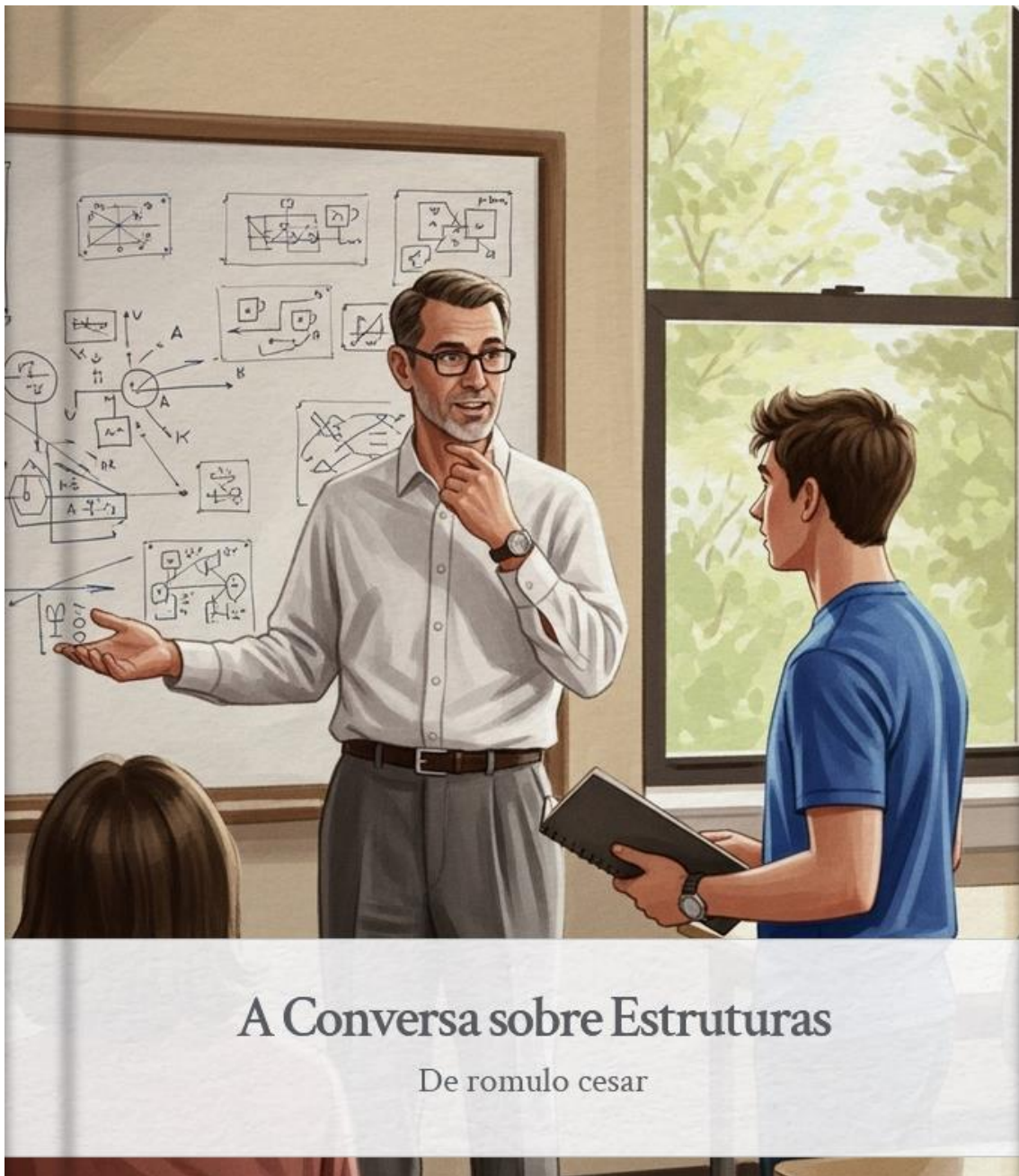
Na criação do projeto também temos algumas considerações:

- Group: Group ID é um identificador único para o grupo de projetos ao qual este artefato pertence. Geralmente, segue a convenção de nomenclatura de pacotes Java reverso(ex: com.suaempresa.seusistema).
 - Ele ajuda a organizar projetos e a evitar conflitos de nomes em repositórios de dependências (como o Maven Central)
- Artifact: Já o Artifact Id é o nome único do seu projeto dentro do grupo. Ele forma o nome base do arquivo JAR ou WAR que será gerado. Juntos o Group Id e o Artifact Id (e a versão) identificam unicamente seu projeto.
- Name: é o nome mais amigável para o seu projeto
- Description: uma descrição do seu projeto
- Package name: é o pacote base para as classes java da aplicação. Ele é derivado do Group Id e Artifact Id, mas pode ser ajustado.
- Packaging: Jar – é o formato padrão para a maioria das aplicações Spring Boot autônomas. Ele inclui todo o código da sua aplicação e suas dependências, vem com um servidor embarcado (Tomcat).
- Java: 17 indica o projeto será desenvolvido e executado usando o Java 17.

Assista o vídeo e realize a prática:



Qual a diferença entre o **Spring Framework** e o **Spring Boot**?

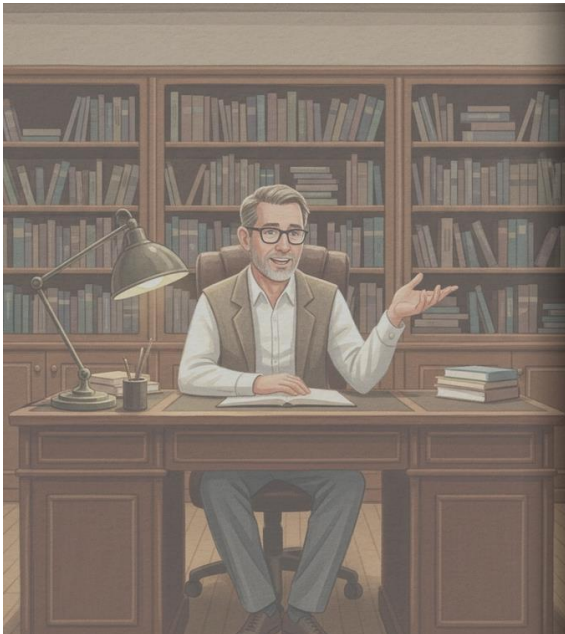


A Conversa sobre Estruturas

De romulo cesar



Leo, um jovem desenvolvedor, aproximou-se da mesa do Dr. Aris Thorne após a aula. "Professor, eu ainda estou um pouco confuso. Qual é exatamente a diferença fundamental entre o Spring Framework e o Spring Boot?"



Dr. Thorne sorriu, um gesto raro e apreciado. "Uma excelente pergunta, Leo. Pense no Spring Framework como uma caixa de ferramentas completa e poderosa. Ele lhe dá todas as peças individuais – vigas, parafusos, engrenagens – para construir quase qualquer tipo de aplicação Java Enterprise."



"Ele opera com base no princípio de Inversão de Controle (IoC) e Injeção de Dependência," continuou o professor. "Isso significa que, em vez de seus objetos criarem suas próprias dependências, o contêiner Spring as 'injeta' para você. Isso promove o baixo acoplamento e facilita os testes."



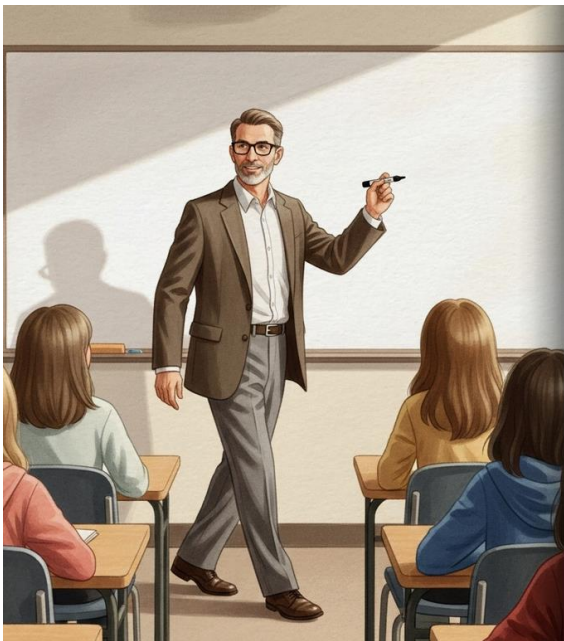
Leo franziu a testa, processando a informação. "Então, eu tenho o poder, mas também a responsabilidade de configurar tudo? O descritor de implantação (web.xml), a configuração do DispatcherServlet, o mapeamento de beans em arquivos XML ou classes de configuração..."



ROMULO CESAR

"Exatamente," confirmou Dr. Thorne.
"O Spring Framework é flexível, mas verboso. Exige uma quantidade significativa de configuração padrão. E é aqui que o Spring Boot entra em cena, não como um substituto, mas como uma evolução."

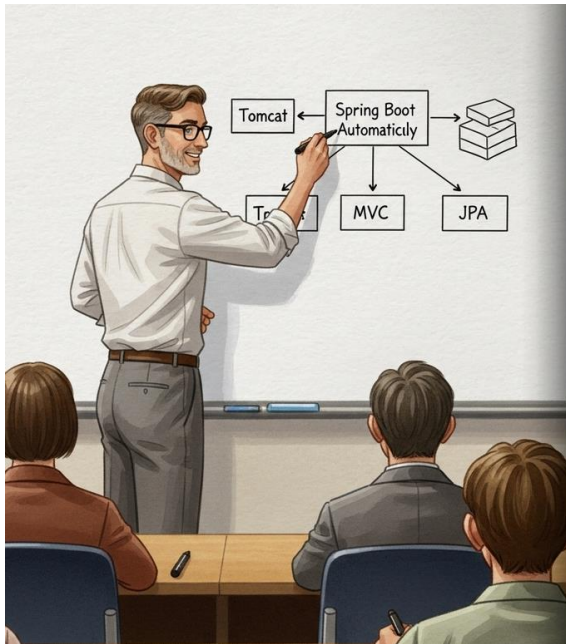
6



ROMULO CESAR

"O Spring Boot", disse o professor, gesticulando com entusiasmo, "olha para essa caixa de ferramentas e diz: 'Eu sei que 90% dos desenvolvedores que constroem uma aplicação web precisam de um servidor web, um framework MVC e uma forma de se conectar a um banco de dados. Vamos pré-montar isso para eles'."

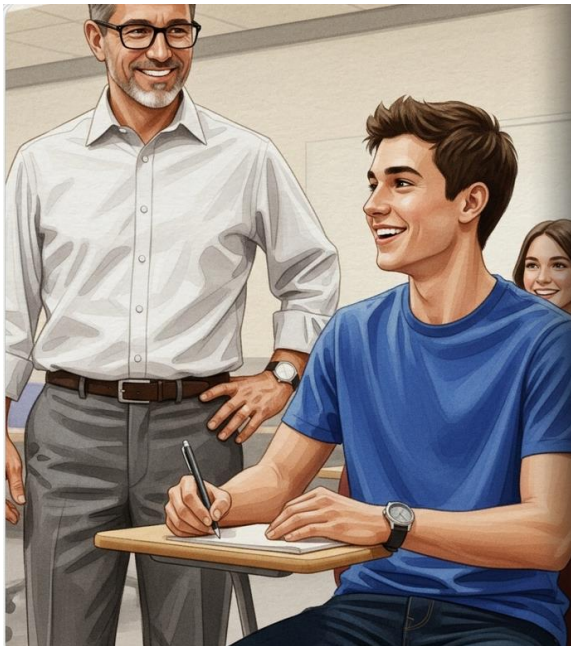
7



"Ele adota uma abordagem 'opinativa'. Ele faz suposições inteligentes sobre as dependências que você provavelmente precisará, com base no que está em seu classpath. Isso é chamado de 'autoconfiguração'. Se ele vê a dependência do Spring MVC, ele configura automaticamente um DispatcherServlet. Se vê o Hibernate, configura uma fonte de dados."



"Então... ele elimina a necessidade de configuração manual?" perguntou Leo, seus olhos se arregalando. "E os 'starters' que eu ouvi falar?"



ROMULO CESAR

"Precisamente! Os 'starter POMs' são convenientes descritores de dependência que você pode incluir em sua aplicação. O spring-boot-starter-web, por exemplo, não só traz o Spring MVC, mas também um servidor Tomcat embutido. Você pode empacotar sua aplicação como um arquivo JAR executável e simplesmente rodá-la, sem precisar de um servidor de aplicação externo."

9



ROMULO CESAR

"Em resumo, Leo," concluiu o Dr. Thorne, "o Spring Framework lhe dá o poder e a flexibilidade para construir qualquer coisa, mas exige que você monte tudo. O Spring Boot usa o Spring Framework para lhe dar uma plataforma de lançamento, pré-configurada e pronta para produção, permitindo que você se concentre na lógica de negócios, não na infraestrutura. Ele não substitui o Spring; ele o torna incrivelmente mais rápido e fácil de usar."

[Recomeçar](#)

10

Leia a história : <https://g.co/gemini/share/24028df9d0e6>

Vamos agora baixar o nosso projeto base:

Modelagem Conceitual

É muito importante aprendermos sobre associações. É extremamente importante para qualquer programador.

- O que são instâncias
- O que é uma associação

Instâncias

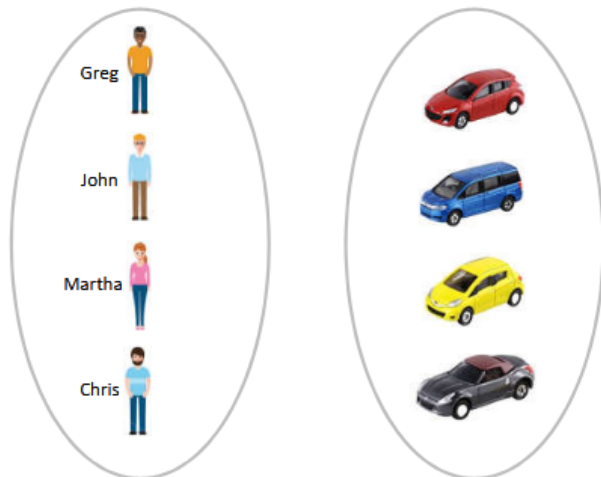
Cada ocorrência dos meus conceitos recebe o nome de INSTÂNCIA ou OBJETO

Exemplo:

Desejo criar um sistema para armazenar informações de pessoas e carros.

Conceitos:

- Pessoa
- Carro



Quem é dono de cada carro?

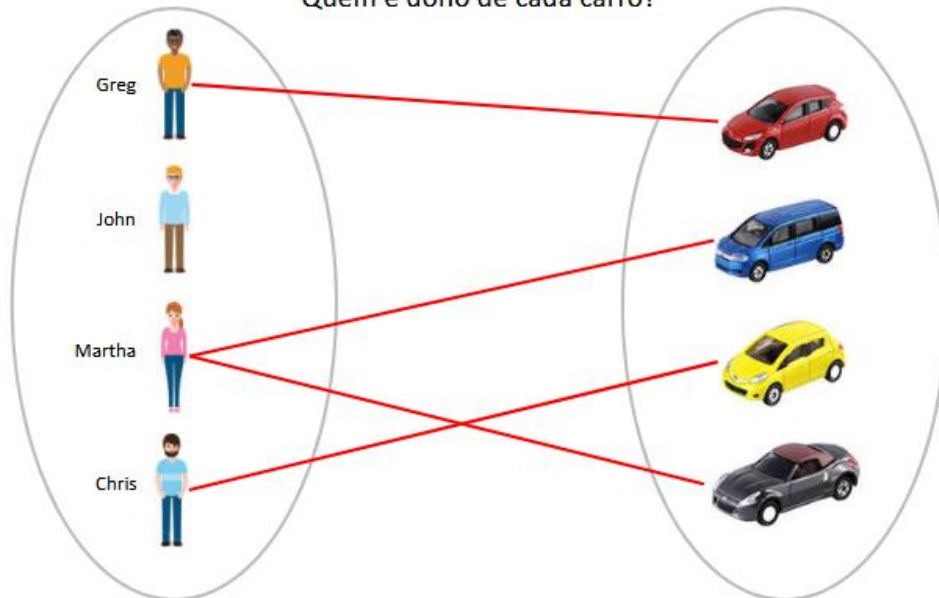
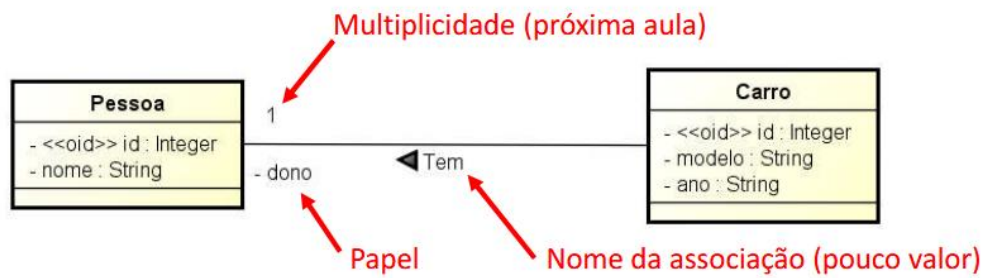
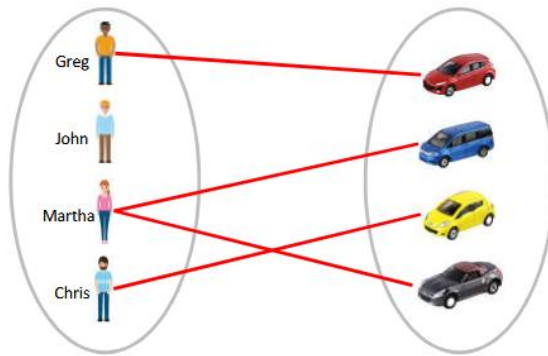


Diagrama UML

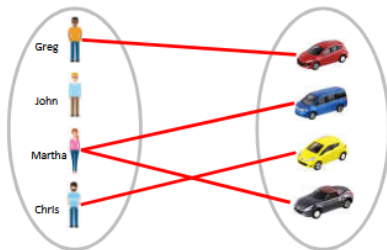


O número 1 (multiplicidade)

- dono é o papel no modelo conceitual

Resumo da aula

- Associação é um relacionamento estático entre dois conceitos

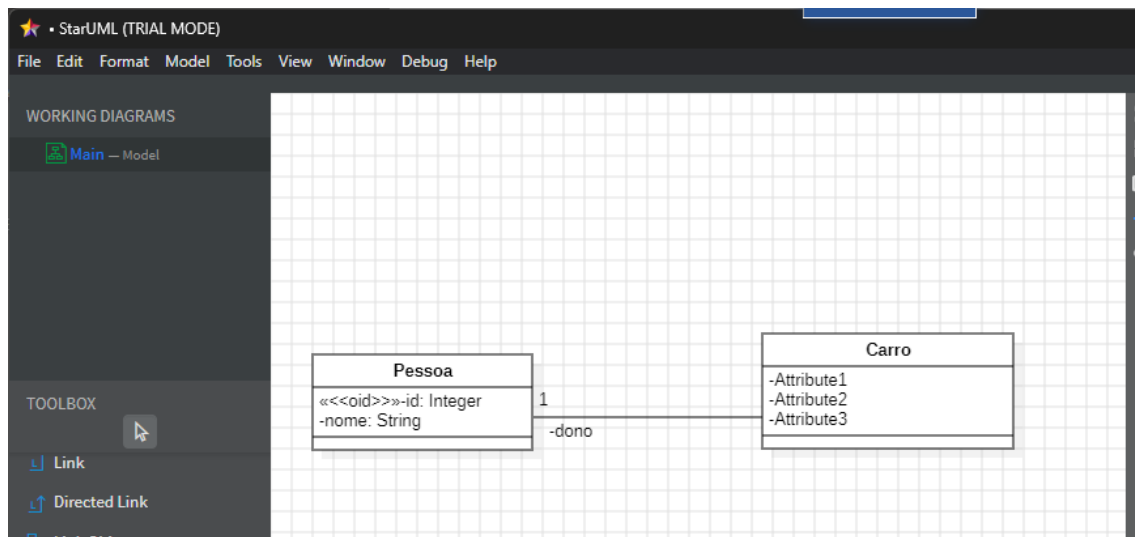


- Nome da associação (pouco valor)
- Nome de papel
- Multiplicidade (próxima aula)

- Não confunda com Modelo Relacional

Exercício:

Agora utilize o StarUML e crie conforme o exemplo dado



Vamos agora pensar no nosso projeto da clínica. Fizemos o paciente e o médico agora falta a consulta.

- Crie um diagrama MER
- Crie um diagrama Conceitual Orientado a Objetos

Você deverá realizar o download do projeto base.

Nosso desafio:

- Deixar funcionando
 - Desenvolvimento da API REST
 - CRUD
 - Validações
 - Paginação e Ordenação
 - Boas práticas REST
 - Tratamento de ERROS
 - Controle de Acesso com JWT
- Continuar agora com o agendamento da consulta.
- Vamos implementar a funcionalidade de agendamento de consultas.
- Vamos aprender sobre regras de negócio. Podemos documentar a API para quem vai consumir nossa API.
- Iremos implementar Testes Automatizados.
- Realizar o Build do projeto

O projeto da clinica médica é o que iremos continuar no mesmo projeto.

Passo a passo para ambientação da aula de hoje no YouTube

