

1. O que preciso fazer?

- Crie um projeto em sua conta GitHub.
- Implemente os desafios descritos no tópico abaixo.
- Use a plataforma Java ou NodeJS para a implementação
- Faça um push para seu repositório com o desafio implementado.
- Envie um email para (recrutamento@tinnova.com.br) avisando que finalizou o desafio com a url do seu projeto.
- Aguarde nosso contato.

2. Lista de exercícios

1) Votos em relação ao total de eleitores

Considerando a tabela abaixo...

total de eleitores = 1000

válidos = 800

votos brancos = 150

nulos = 50

Faça uma classe com 3 métodos que calculam...

- o percentual do votos válidos em relação ao total de eleitores,
- o percentual de brancos em relação ao total de eleitores
- o percentual de nulos em relação ao total de eleitores.

Dica: “em relação ao total” significa que você deve dividir, por exemplo, “nulos” pelo total de eleitores, válidos pelo total de eleitores, etc...

Utilize programação orientada a objetos.

2) Algoritmo de ordenação Bubble Sort

Imagine o seguinte vetor.

```
v = {5, 3, 2, 4, 7, 1, 0, 6}
```

Faça um algoritmo que ordene o vetor acima utilizando o **Bubble Sort**.

O Bubble Sort ordena de par em par. Ele pega os dois primeiros elementos e pergunta se o primeiro é maior que o segundo. Se sim, os elementos são trocados (swap), se não, são mantidos. Vai repetindo o processo até o final do vetor.

Obviamente que ele não consegue ordenar todo o vetor em uma única rodada, ele terá que passar pelo vetor um certo número de vezes.

De maneira mais formal podemos destacar:

1. Percorra o vetor inteiro comparando elementos adjacentes (dois a dois)
2. Troque as posições dos elementos se eles estiverem fora de ordem
3. Repita os dois passos acima ($n - 1$) vezes, onde n é igual ao tamanho do vetor

OK, vamos fazer um exemplo para facilitar o entendimento.

Voltemos ao nosso vetor.

```
5, 3, 2, 4, 7, 1, 0, 6
```

Sabemos que iremos repetir o vetor $n - 1$ vezes. O tamanho do vetor é 8, logo iremos repetir 7 vezes o vetor (8-1).

Vamos chamar cada repetição de iteração.

Então, na **primeira iteração**, pegamos os dois primeiros valores e trocamos se estiverem fora de ordem.

```
(5 3) 2 4 7 1 0 6  pegamos o primeiro par
3--5  2 4 7 1 0 6  trocamos
```

```
3 (5 2) 4 7 1 0 6  pegamos o próximo par
3 2--5  4 7 1 0 6  trocamos
```

```
3 2 (5 4) 7 1 0 6  pegamos o próximo par
3 2 4--5  7 1 0 6  trocamos
```

```
3 2 4 (5 7) 1 0 6  pegamos o próximo par
3 2 4 5--7  1 0 6  mantemos <----
```

```
3 2 4 5 (7 1) 0 6  pegamos o próximo par
```

```
3 2 4 5 1--7 0 6 trocamos
```

```
3 2 4 5 1 (7 0) 6 pegamos o próximo par
```

```
3 2 4 5 1 0--7 6 trocamos
```

```
3 2 4 5 1 0 (7 6) pegamos último par
```

```
3 2 4 5 1 0 6 7 trocamos
```

Chegamos ao fim da primeira iteração e, como dito, não foi suficiente para ordenar o vetor.

Teremos que reiniciar, só que agora sabemos que, pelo menos, o último valor (7) já está em seu devido lugar

Então iremos marcá-lo e não precisaremos percorrer todo o vetor na segunda iteração.

```
3 2 4 5 1 0 6 [7]
```

Esse detalhe é importante e fará toda a diferença no entendimento do algoritmo.

Todo esse processo se repetirá até que todos os itens estejam devidamente ordenados.

3) Fatorial

Faça um programa que calcule o fatorial de um número qualquer.

Vamos lembrar o que é o fatorial?

Seja n um número natural, tal que $n \geq 2$, chama-se fatorial de n o produto de todos os números naturais consecutivos de n até 1.

Por exemplo,

$$5! = 1 \times 2 \times 3 \times 4 \times 5 = 120$$

Veja mais alguns resultados e que você poderá utilizar como testes:

$$0! = 1$$

$$1! = 1$$

$$2! = 2$$

$$3! = 6$$

$$4! = 24$$

$$5! = 120$$

$$6! = 720$$

Atente que $0! = 1$ porque o produto vazio (produto de nenhum número) é 1.

$$fatorial(n) = \begin{cases} 1 & \text{se } n = 0 \\ n \times fatorial(n - 1) & \text{se } n > 0 \end{cases}$$

4) Soma dos múltiplos de 3 ou 5.

Fazer uma implementação que faça a soma de todos os números que sejam múltiplos de 3 ou 5.

Se listar todos os números naturais abaixo de 10 que são múltiplos de 3 ou 5, ficamos com 3, 5, 6 e 9. A soma desses múltiplos é de 23.

A implementação deve ser capaz de receber por parametro um número X se já retornado a soma de todos os números múltiplos de 3 ou 5.

5) Cadastro de veículos

Criar uma aplicação back-end (Java ou NodeJS) baseada em web services usando JSON.

Deverá haver um front-end em modo Single Page Application que se comunique com os serviços criados no back-end.

Requisitos:

- Permitir o cadastro de veículos
- Permitir a atualização de dados de um veículo
- Permitir a exclusão de um veículo
- Exibir a informação de quantos veículos estão como não vendidos na base.
- Exibir a informação da distribuição de veículos por década de fabricação
Exemplo:
 - Década 1990 -> 15 veículos
 - Década 2000 -> 31 veículos
- Exibir a informação da distribuição de veículos por fabricante.
Exemplo:
 - Ford -> 14 veículos
 - Honda -> 8 veículos
- Exibir os carros registrados durante a última semana.
- Deverá haver consistência das marcas fornecidas. Não poderá haver marcas escritas de forma errada (Exemplo: Volksvagen, Forde, Xevrolé, etc. não serão aceitos no cadastro)

Missão

Desenvolver uma API JSON RESTful, que utilize todos os métodos (GET, POST, PUT, PATCH, DELETE).

Faça o teste unitário da API

Especificação

Monte uma base de veículo com a seguinte estrutura:

```
veiculo:  string
marca:    string
ano:      integer
descricao: text
vendido:  bool
created:  datetime
updated:  datetime
```

API endpoints

GET /veiculos

Retorna todos os veículos

GET /veiculos?marca={marca}&ano={ano}&cor={cor}

Retorna todos os veículos de acordo com os parâmetros passados

GET /veiculos/{id}

Retorna os detalhes do veículo

POST /veiculos

Adiciona um novo veículo

PUT /veiculos/{id}

Atualiza os dados de um veículo

PATCH /veiculos/{id}

Atualiza apenas alguns dados do veículo

DELETE /veiculos/{id}

Apaga o veículo

3. Critérios de avaliação

- Facilidade de configuração do projeto
- Performance
- Código limpo e organização
- Documentação de código
- Documentação do projeto (readme)
- Boas práticas de desenvolvimento
- Design Patterns