**Backend para gerenciar times de futebol**

Você é responsável por construir o backend de um novo gerenciador de times de futebol. Após fecharem o escopo do projeto, você e sua equipe definiram a @interface que o software deve implementar. A interface é a seguinte :

public interface MeuTimeInterface {

@Desafio("incluirTime")

public void incluirTime(Long id, String nome, LocalDate dataCriacao, String corUniformePrincipal, String corUniformeSecundario);

@Desafio("incluirJogador")

public void incluirJogador(Long id, Long idTime, String nome, LocalDate dataNascimento, Integer nivelHabilidade, BigDecimal salario);

@Desafio("definirCapitao")

public void definirCapitao(Long idJogador);

@Desafio("buscarCapitaoDoTime")

public Long buscarCapitaoDoTime(Long idTime);

@Desafio("buscarNomeJogador")

public String buscarNomeJogador(Long idJogador);

@Desafio("buscarNomeTime")

public String buscarNomeTime(Long idTime);

@Desafio("buscarMaiorSalario")

public Long buscarJogadorMaiorSalario(Long idTime);

@Desafio("buscarSalarioDoJogador")

public BigDecimal buscarSalarioDoJogaodor(Long idJogador);

@Desafio("buscarJogadoresDoTime")

public List<Long> buscarJogadoresDoTime(Long idTime);

@Desafio("buscarMelhorJogadorDoTime")

public Long buscarMelhorJogadorDoTime(Long idTime);

@Desafio("buscarJogadorMaisVelho")

public Long buscarJogadorMaisVelho(Long idTime);

@Desafio("buscarTimes")

public List<Long> buscarTimes();

@Desafio("buscarTopJogadores")

public List<Long> buscarTopJogadores(Integer top);

}

Os dados devem ficar armazenados na memória.

**Requisitos de Sistema**

* Parâmetros com \* são obrigatórios.
* Os parâmetros, com exceção de identificadores, são sempre íntegros e farão sentido.

**@Desafio(“incluirTime)”**

Realiza a inclusão de um novo time.

* Long id\* Identificador do time
* String nome\* Nome do Time
* LocalDate dataCriacao\* Data de criação do time
* String corUniformePrincipal\* Cor do uniforme principal do time
* String corUniformeSecundario\* Cor do uniforme secundário do time

Exceções:

* Caso o identificador já exista, retornar br.com.codenation.desafio.exceptions.IdentificadorUtilizadoException

**@Desafio(“incluirJogador)”**

Realiza a inclusão de um novo jogador.

* Long id\* Identificador do Jogador
* Long idTime\* Identificador do time
* String nome\* Nome do Jogador
* LocalDate dataNascimento\* Data de nascimento do Jogador
* Integer nivelHabilidade\* Nível de habilidade do jogador (0 a 100)
* BigDecimal salario\* Salário do jogador

Exceções:

* Caso o identificador já exista, retornar br.com.codenation.desafio.exceptions.IdentificadorUtilizadoException
* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“definirCapitao”)**

Define um jogador como capitão do seu time. Um time deve ter apenas um capitão, por tanto o capitão anterior voltará a ser apenas jogador.

* Long idJogador\* Identificador do jogador.

Exceções:

* Caso o jogador informado não exista, retornar br.com.codenation.desafio.exceptions.JogadorNaoEncontradoException

**@Desafio(“buscarCapitaoDoTime”)**

Mostra o identificador do capitão do time.

* Long idTime\* Identificador do Time

Exceções:

* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

Caso o time informado não tenha um capitão, retornar br.com.codenation.desafio.exceptions.CapitaoNaoInformadoException

**@Desafio(“buscarNomeJogador”)**

Retorna o nome do jogador.

* Long idJogador\* Identificador do jogador

Exceções

Caso o jogador informado não exista, retornar br.com.codenation.desafio.exceptions.JogadorNaoEncontradoException

**@Desafio(“buscarNomeTime”)**

Retorna o nome do time.

* Long idTime\* Identificador do Time

Exceções

* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“buscarJogadoresDoTime”)**

Retorna a lista com o identificador de todos os jogadores do time, ordenada pelo id.

* Long idTime\* Identificador do Time

Exceções

* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“buscarMelhorJogadorDoTime”)**

Retorna o identificador do melhor jogador do time.

* Long idTime\* Identificador do time.

Exceções:

* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“buscarJogadorMaisVelho”)**

Retorna o identificador do jogador mais velho do time. Usar o menor identificador como critério de desempate.

* Long idTime\* Identificador do time
* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“buscarTimes”)**

Retorna uma lista com o identificador de todos os times cadastrado, ordenada pelo identificador. Retornar uma lista vazia caso não encontre times cadastrados.

**@Desafio(“buscarJogadorMaiorSalario”)**

Retorna o identificador do jogador com maior salário do time. Usar o menor identificador como critério de desempate.

* Long idTime\* Identificador do time.

Exceções:

* Caso o time informado não exista, retornar br.com.codenation.desafio.exceptions.TimeNaoEncontradoException

**@Desafio(“buscarSalarioDoJogador”)**

Retorna o salário do jogador.

* Long idJogador\* Identificador do jogador

Exceções:

* Caso o jogador informado não exista, retornar br.com.codenation.desafio.exceptions.JogadorNaoEncontradoException

**@Desafio(“buscarTopJogadores”)**

Retorna uma lista com o identificador dos top melhores jogadores, utilizar o menor identificador como critério de desempate.

* Integer top\* Quantidade de jogares na lista

Exceções:

* Caso não exista nenhum jogador cadastrado, retornar uma lista vazia.

**@Desafio(“buscarCorCamisaTimeDeFora”)**

Retorna a cor da camisa do time adversário. Caso a cor principal do time da casa seja igual a cor principal do time de fora, retornar cor secundária do time de fora. Caso a cor principal do time da casa seja diferente da cor principal do time de fora, retornar cor principal do time de fora.

* Long idTimeDaCasa\* Identificador do time da casa
* Long idTimeDeFora\* Identificador do time de fora

**Tópicos**

Neste desafio você aprenderá:

* Variáveis e métodos
* Operadores matemáticos
* Controle de fluxo
* Listas
* Tratamento de exceções

**Requisitos**

​ ​Para este desafio você precisará:

* Java 8 +

Para instalar no MacOS, usando o HomeBrew basta executar:

brew cask install java

Para instalar no Linux e Windows confira os links na sessão de conteúdo