

Aula - Classes e Encapsulamento

Materiais Complementares

- ▶ PT
- ▶ EN

Materiais de Aula

- ▶ 📄 Slides
- ▶ 📄 Codando Junto
- ▼ 📄 Exercícios feitos em aula
 - 📎 AulaClassesEncapsulamento.rar 16,8KB
- ▶ 🎧 Gravação da aula

Instruções gerais

Hoje vamos praticar sobre POO - Programação Orientada a Objetos. Resumindo, utilizamos classes para abstrair algo da vida real para a linguagem de programação. Pode-se dizer que classes tem importante papel em "modelar" nossos dados na aplicação, representando como essa entidade é na vida real.

Inicie o projeto por aqui

⚠ IMPORTANTE ⚠

- 1) Crie uma branch a partir da branch master para trabalhar no exercício de hoje. O nome da branch de hoje deve ser: `classes-encapsulamento`
- 2) Dentro da pasta do módulo atual, crie uma pasta chamada `classes-encapsulamento` para trabalhar no exercício de hoje

🔧 Quero iniciar o projeto do zero, como faço?

▼ Veja aqui

1. Dentro da pasta do projeto, criar a pasta `src` e dentro dessa pasta, criar o arquivo em Typescript chamado `index.ts`

2. Criar o `package.json`

📄 Comando: `npm init -y` (o -y cria um package.json com configurações padrão)

3. Instalar o `typescript` como *dev dependencies*

📄 Comando: `npm i typescript -D`

4. Criar o `tsconfig.json`

📄 Comando: `npx tsc --init`

```
// dentro do tsconfig.json, descomentar e atribuir valores às propriedades:  
  
"target": "es6",
```

```
{...}
"sourceMap": true,
{...}
"outDir": "./build",
"rootDir" : "./src",
{...}
"removeComments": true,

// target vem por padrão como es5
// outDir especifica a pasta em que serão salvos os arquivos transpilados
// rootDir especifica a pasta raiz do projeto
// removeComments remove comentários dos arquivos transpilados
```

Quero iniciar através do template disponibilizado, como faço?

► [Veja aqui](#)


Enunciado

Hoje vamos praticar a aplicação de POO - Programação Orientada a Objetos. Baixe o template abaixo para iniciar a atividade, ele possui o código do gabarito do projeto labecommerce.

Antes de iniciar os exercícios, rode o `npm install`, o `npm run migrations` e edite o arquivo `.env`!

Para rodar o servidor é necessário executar o `npm run build` antes do `npm run start`. Outra saída é rodar o comando `npm run dev` para facilitar o processo de desenvolvimento do código.

Template

 [template-classes-encapsulamento.zip](#) 10.5KB

Exercícios propostos

Como apresentado em aula, os exercícios de hoje abordarão o refatoramento do código para a aplicação de um paradigma, o famoso POO. A ideia é demonstrar que o uso de um paradigma não influencia no resultado final e sim em como elaboramos o código.

Exercício 1

Modifique o type `User` para que ele se torne uma classe. Lembre-se também de refatorar os códigos onde o type era utilizado, pois com classes é necessário instanciar o objeto.

Exercício 2

Agora modifique o type `Product` para que ele também seja uma classe. Novamente, lembre-se de editar a parte do código onde o type era utilizado para que os objetos sejam instanciados corretamente. Referencie o código da aula para implementar o `TProduct` e utilizá-lo no `data.ts`.

Exercício 3

Por fim, crie a classe `Purchase` no lugar de seu type. Não se preocupe com o refatoramento e uso do `PurchaseDB`. Finalize a implementação com o uso de instâncias onde o type `Purchase` era utilizado.

Dicas Gerais

Pratique o uso de encapsulamento (`public/private`) junto com os `getters` e `setters`!

Teste os endpoints afetados pela mudança antes de partir para o próximo exercício!

Procure evitar ao máximo editar código que não seja necessário para a implementação do POO. Em caso de dúvidas utilize o canal no slack! Sua pergunta pode ajudar outras pessoas a entenderem o problema!