

## **CASOS DE TESTE**

São Paulo,

2024

## SUMÁRIO

<b>1 Cenário 1 – Criando um novo produto .....</b>	<b>3</b>
1.1 Caso de teste 1 – cadastro realizado com sucesso .....	3
1.2 Caso de teste 2– Falha no cadastro (Falta de descrição do produto).....	5
1.3 Caso de teste 3– Falha no cadastro (Falta de descrição e preço do produto) .....	6
1.4 Caso de teste 4– Falha no cadastro (Preço enviado como string).....	7
<b>2 Cenário 2 – Lendo detalhes do produto .....</b>	<b>8</b>
2.1 Caso de teste 1– Visualização realizada com sucesso .....	8
2.2 Caso de teste 2– Produto não possui descrição .....	8
2.3 Caso de teste 3– Preço do produto com valor negativo).....	9
<b>3 Cenário 3 – Atualizando produto existente .....</b>	<b>10</b>
3.1 Caso de teste 1– Atualização realizada com sucesso.....	10
<b>4 Cenário 4 – Deletando um produto.....</b>	<b>11</b>
4.1 Caso de teste 1– Produto deletado com sucesso .....	11

## 1. CENÁRIO 1 – Criando um novo produto

- Preencher o formulário de produto com “Produto A”, “Descrição do Produto A” e “Preço do Produto A”.
- Clicar no botão salvar.
- Visualizar “Produto A” na lista de produtos

**OBSERVAÇÕES:** Segundo a documentação Swagger da API, todos os campos são obrigatórios na inclusão:

```
CreateProductDto {  
  title*           string  
  price*           number  
  description*     string  
  categoryId*      number  
  images*          string  
}
```

Então, o usuário precisará informar também o Id categoria e a imagem do produto.

### 1.1 Caso de teste 1- Cadastro realizado com sucesso

**PREMISSA:** Cenário onde os cadastros são realizados com sucesso, todas as informações são preenchidas corretamente, salva os produtos clicando no botão salvar e o visualiza na lista de produtos.

➤ Produto de exemplo contido no JSON enviado:

```
{  
  "title": "Produto A",  
  "price": 10,  
  "description": "Descrição do produto A",
```

```
"categoryId": 1,
"images": ["https://placeimg.com/640/480/any"]
}
```

Response esperado:

```
{
  "title": "Produto A",
  "price": 10,
  "description": "Descrição do produto A",
  "images": [
    ["https://placeimg.com/640/480/any\""]
  ],
  "category":
  {
    "id": 1,
    "name": "soy el 9",
    "image": "https://i.imgur.com/Qkla5tT.jpeg",
    "creationAt": "2024-09-26T14:10:18.000Z",
    "updatedAt": "2024-09-26T22:08:20.000Z"
  },
  "id": 96,
  "creationAt": "2024-09-27T13:13:24.000Z",
  "updatedAt": "2024-09-27T13:13:24.000Z"
}
```

➤ Ao visualizar a lista de produtos, espera-se encontrar:

```
{
  "id": 96,
```

```

"title": "Produto A",
"price": 10,
"description": "Descrição do produto A",
"images": [
  ["https://placeimg.com/640/480/any\""],
],
"creationAt": "2024-09-27T13:13:24.000Z",
"updatedAt": "2024-09-27T13:13:24.000Z",
"category": {
  "id": 1,
  "name": "soy el 9",
  "image": "https://i.imgur.com/Qkla5tT.jpeg",
  "creationAt": "2024-09-26T14:10:18.000Z",
  "updatedAt": "2024-09-26T22:08:20.000Z"
}
}

```

➤ Cypress deve retornar:

**Log** Todos os produtos foram adicionados com sucesso!

## 1.2 Caso de Teste 2 - Falha no cadastro (Falta de descrição do produto)

**PREMISSA:** Cenário onde no cadastro dos produtos todas as informações são preenchidas corretamente, porém o campo “description” de um produto ficou vazio. Ao clicar no botão, a API deve retornar um erro, indicando que não foi possível salvar o cadastro desse produto pois “description” não pode ser vazio, já que é um campo obrigatório.

➤ Produto do JSON sem descrição enviado:

```
{
  "title": "Produto J",
  "price": 100,
  "categoryId": 1,
  "images": ["https://placeimg.com/640/480/any"]
}
```

➤ Response esperado:

**Request**POST 400 https://api.escuelajs.co/api/v1/products/

Assertexpected [ Array(2) ] to include \*\*description should not be empty\*\*

**Log** Erro esperado no cadastro do produto "Produto J": description should not be empty, description must be a string

**Log** Erro no cadastro do produto "Produto J" foi encontrado e tratado com sucesso.

### 1.3 Caso de teste 3: Falha no Cadastro (Falta de descrição e preço do produto)

**PREMISSA:** Cenário onde no cadastro dos produtos todas as informações são preenchidas corretamente, porém os campos “description” e “price” de um produto ficaram vazios. Ao clicar no botão, a API deve retornar um erro, indicando que não foi possível salvar o cadastro desse produto pois “description” e “price” não podem ser vazios, já que é um campo obrigatório.

➤ JSON enviado:

```
{
  "title": "Produto J",
  "categoryId": 1,
  "images": ["https://placeimg.com/640/480/any"]
}
```

➤ Resultado esperado:

**Request** POST 400 <https://api.escuelajs.co/api/v1/products/>

Assert expected [ Array(4) ] to satisfy [Function]

**Log** Erro esperado no cadastro do produto "Produto J": price should not be empty, price must be a positive number, description should not be empty, description must be a string

**Log** Erro no cadastro do produto "Produto J" foi encontrado e tratado com sucesso.

**Log** Produto "Produto J" não foi adicionado, conforme esperado.

#### 1.4 Caso de teste 4 – Falha no Cadastro (Preço enviado como string)

**PREMISSA:** Cenário onde no cadastro todas as informações são preenchidas corretamente, porém o campo “price” do “Produto Z” recebe um valor string ao invés de number. Ao clicar no botão, API deve retornar um erro, indicando que não foi possível salvar o cadastro do “Produto Z” pois “price” não podem ser do tipo string, já que a API espera receber apenas tipo number nesse campo.

➤ Produto com problema no JSON enviado:

```
{
  title: "Produto Z",
  price: "10",
  description: "Descrição do produto Z",
  categoryId: 1,
  images: ["https://placeimg.com/640/480/any"]
}
```

➤ Resultado esperado

**Log** Erro esperado: o preço "10" está em formato string, mas deveria ser number

**Log** Produto "Produto Z" não será cadastrado devido ao preço estar como string.

**Log** Produto "Produto Z" não foi adicionado, conforme esperado.

## 2. CENÁRIO 2 – Lendo detalhes do produto

- Clicar no link detalhes do “Produto A”
- Visualizar os detalhes do “Produto A”

**PREMISSA GERAL:** Espera-se que a API tenha um tratamento para receber valores nos campos, como por exemplo não receber strings parcialmente vazias como “ ”, ou números negativos.

### 2.1 Caso de teste 1 – Visualização realizada com sucesso

**PREMISSA:** Nesse cenário, após realizar o cadastro do “Produto A”, ao clicar no botão de visualizar detalhes, deve ser possível encontrar a descrição do produto.

➤ Produto A no JSON enviado:

```
{  
  title: "Produto A",  
  price: 10,  
  description: "Descrição do produto A",  
  categoryId: 1,  
  images: ["https://placeimg.com/640/480/any"]  
};
```

➤ Resultado esperado:

**Log** Detalhes do produto A: Descrição do produto A

**Log** Descrição visualizada com sucesso!



## 2.2 Caso de teste 2 - Produto não possui descrição

**PREMISSA:** Nesse cenário, cadastramos o campo “description” do “Produto Z” apenas com um espaço no texto de forma que ele fique assim: “ ”. Após realizar o cadastro do “Produto Z”, espera-se que a API retorne um json com a descrição “ ”, e é um erro que precisa ser tratado na construção da API:

- Produto Z no JSON enviado:

```
{
  title: "Produto Z",
  price: 10,
  description: " ",
  categoryId: 1,
  images: ["https://placeimg.com/640/480/any"]
};
```

- Resultado esperado:

**Log** Descrição do produto Z:

**Log** A descrição do Produto Z está vazia ou contém apenas espaços.

## 2.3 Caso de teste 3 – Preço do produto com valor negativo

**PREMISSA :** Ao realizar o cadastro do “Produto A” com valor negativo, espera-se que o Cypress não cadastre o “Produto A” e nos informe em um log

- Produto A no JSON de produtos enviado:

```
{
  title: "Produto A",
  price: -10,
  description: "Descrição do produto A",
  categoryId: 1,
```

```
images: ["https://placeimg.com/640/480/any"]
}
```

➤ Resultado esperado:

**Log** Produto "Produto A" não foi adicionado, pois o preço está incorreto: -10

### 3. CENÁRIO 3 – Atualizando produto existente

- Alterar nome do produto “Produto A” para “Produto B”.
- Clicar no botão de salvar.
- Visualizar “Produto B” na lista de produtos.

#### 3.1 Caso de teste 1 – Atualização realizada com sucesso

**PREMISSA:** Ao entrar na página de edição do “Produto A”, depois de alterar o nome para “Produto B”, espera-se poder visualizar “Produto B” na lista de produtos.

➤ JSON enviado:

```
body:

{
  title: "Produto B"
}
```

➤ Resultado esperado:

**Log** Título alterado e verificado com sucesso: Produto B

### 4. CENÁRIO 4 – Deletando um produto

- Visualizar lista de produtos.
- Clicar no botão Deletar do “Produto B”.
- Não visualizar “Produto B” na lista de produtos.

#### 4.1 Caso de teste 1 – Produto deletado com sucesso

**PREMISSA:** Nesse cenário, após visualizar “Produto B”, clicar no botão de deletar do “Produto B”. Assim, espera-se não visualizar o “Produto B” na lista de produtos.

Dessa forma, um log deve ser enviado como resposta:

**Log** Produto B deletado com sucesso!

**Log** Confirmação: Produto B não está mais na lista.