Massa de dados utilizados nos testes.

1. Cenário - Cadastrar novo pedido de pet com sucesso

### Detalhes da API:

\*\*Endpoint\*\*: https://petstore.swagger.io/v2/store/order

\*\*Método\*\*: POST

\*\*Parâmetros de Entrada\*\*:

- `id`: Integer (ID do pedido)

- `petId`: Integer (ID do pet)

- `quantity`: Integer (quantidade de pets)

- `shipDate`: String (data de envio no formato ISO 8601)

- `status`: String (status do pedido,

pode ser: "placed",

"approved",

"delivered")

- `complete`: Boolean (determina se o pedido está completo)

\*\*Exemplo de Entrada\*\*:

```json

{

"id": 12345,

"petId": 1,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed",

"complete": true

}

```

\*\*Exemplo de Resposta de Sucesso\*\*:

```json

{

"id": 12345,

"petId": 1,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed",

"complete": true

}

```

### Casos de Teste:

\*\*1.

Teste de cadastro de pedido com sucesso:\*\*

- \*\*Passo a Passo:\*\*

1. Configurar a requisição POST para o endpoint `https://petstore.swagger.io/v2/store/order`.

1. Enviar o payload com todos os dados corretamente preenchidos:

```json

{

"id": 12345,

"petId": 1,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed",

"complete": true

}

```

1. Verificar se a resposta tem o status de 200 OK.
2. Verificar se a resposta JSON corresponde ao payload enviado.

\*\*2.Teste de cadastro com campo obrigatório faltando (por exemplo,

`petId`):\*\*

- \*\*Passo a Passo:\*\*

1. Configurar a requisição POST para o endpoint.
2. Enviar o payload sem o campo `petId`:

```json

{

"id": 12345,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed",

"complete": true

}

```

1. Verificar se a resposta tem o status de 400 Bad Request.
2. Verificar a mensagem de erro no corpo da resposta.

\*\*3. Teste de envio com `status` inválido:\*\*

- \*\*Passo a Passo:\*\*

1. Configurar a requisição POST para o endpoint.

2. Enviar o payload com `status` inválido:

```json

{

"id": 12345,

"petId": 1,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "invalid\_status",

"complete": true

}

```

3. Verificar se a resposta tem o status de 400 Bad Request.

4. Verificar a mensagem de erro no corpo da resposta.

\*\*4. Teste de cadastro com campo `quantity` como string:\*\*

- \*\*Passo a Passo:\*\*

1. Configurar a requisição POST para o endpoint.

2. Enviar o payload com `quantity` como string:

```json

{

"id": 12345,

"petId": 1,

"quantity": "dois",

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed",

"complete": true

}

```

3. Verificar se a resposta tem o status de 400 Bad Request.

4. Verificar a mensagem de erro no corpo da resposta.

\*\*5. Teste de envio sem o campo `complete`:\*\*

- \*\*Passo a Passo:\*\*

1. Configurar a requisição POST para o endpoint.

2. Enviar o payload sem o campo `complete`:

```json

{

"id": 12345,

"petId": 1,

"quantity": 2,

"shipDate": "2023-10-10T23:22:56.

000Z",

"status": "placed"

}

```

3. Verificar se a resposta tem o status de 200 OK.

1. Verificar se o campo `complete` no corpo da resposta é `false` por padrão.