Massa de dados para os testes no endpoint:

Atualizar dados de um pet existente (PUT /pet)

### Caso de Teste 1: Atualizar dados do pet com sucesso

\*\*Descrição\*\*: Atualizar os dados de um pet existente com informações válidas.

\*\*Passos\*\*:

1. Configure um ambiente de teste com a URL https://petstore.

swagger.

io/v2/pet.

1. Prepare os dados para atualizar o pet,

incluindo um ID existente e novos detalhes do pet (nome,

categoria,

status,

etc.

).

\*\*Chamada da API\*\*:

```json

PUT https://petstore.

swagger.

io/v2/pet

Content-Type: application/json

{

"id": 12345,

"name": "Rex",

"category": {

"id": 1,

"name": "Dogs"

},

"status": "available"

}

```

1. Envie a solicitação PUT.
2. Verifique a resposta da API:

\*\*Expectativa\*\*:

```json

{

"id": 12345,

"name": "Rex",

"category": {

"id": 1,

"name": "Dogs"

},

"status": "available"

}

```

### Caso de Teste 2: Atualizar dados do pet inexistente

\*\*Descrição\*\*: Tentar atualizar um pet que não existe na base de dados.

\*\*Passos\*\*:

1. Configure um ambiente de teste com a URL https://petstore.

swagger.

io/v2/pet.

1. Prepare os dados para atualizar um pet com um ID inexistente.

\*\*Chamada da API\*\*:

```json

PUT https://petstore.

swagger.

io/v2/pet

Content-Type: application/json

{

"id": 99999,

"name": "Fido",

"category": {

"id": 2,

"name": "Cats"

},

"status": "sold"

}

```

1. Envie a solicitação PUT.
2. Verifique a resposta da API:

\*\*Expectativa\*\*: Status code 404 (Not Found)

```json

{

"code": 404,

"type": "error",

"message": "Pet not found"

}

```

### Caso de Teste 3: Atualizar dados com payload malformado

\*\*Descrição\*\*: Enviar uma solicitação PUT com um corpo de requisição malformado.

\*\*Passos\*\*:

1.

Configure um ambiente de teste com a URL

https://petstore.swagger.io/v2/pet.

2. Prepare um payload malformado.

\*\*Chamada da API\*\*:

```json

PUT https://petstore.

swagger.

io/v2/pet

Content-Type: application/json

{

"id": "12A34B",

"name" "Spike",

"category": {

"id": 3,

"nae": "Birds"

},

"status": "unavailable"

}

```

3. Envie a solicitação PUT.

4. Verifique a resposta da API:

\*\*Expectativa\*\*: Status code 400 (Bad Request)

```json

{

"code": 400,

"type": "error",

"message": "Bad Request"

}

```

### Caso de Teste 4: Atualizar dados sem campo obrigatório (ID)

\*\*Descrição\*\*: Enviar uma solicitação PUT sem o campo obrigatório ID.

\*\*Passos\*\*:

1.Configure um ambiente de teste com a URL https://petstore.swagger.io/v2/pet.

2.Prepare o corpo do pedido sem o campo ID.

\*\*Chamada da API\*\*:

```json

PUT https://petstore.

swagger.

io/v2/pet

Content-Type: application/json

{

"name": "Luna",

"category": {

"id": 4,

"name": "Rabbits"

},

"status": "sold"

}

```

3.Envie a solicitação PUT.

4.Verifique a resposta da API:

\*\*Expectativa\*\*: Status code 400 (Bad Request)

```json

{

"code": 400,

"type": "error",

"message": "Invalid input"

}

```

### Caso de Teste 5: Atualizar dados com status inválido

\*\*Descrição\*\*: Atualizar um pet com status inválido (não suportado pela API).

\*\*Passos\*\*:

1. Configure um ambiente de teste com a URL https://petstore.swagger.io/v2/pet.
2. Prepare um payload com um status inválido.

\*\*Chamada da API\*\*:

```json

PUT https://petstore.

swagger.

io/v2/pet

Content-Type: application/json

{

"id": 12345,

"name": "Buddy",

"category": {

"id": 1,

"name": "Dogs"

},

"status": "infected"

}

```

3.Envie a solicitação PUT.

4.Verifique a resposta da API:

\*\*Expectativa\*\*: Status code 400 (Bad Request)

```json

{

"code": 400,

"type": "error",

"message": "Invalid status value"

}

```