RELATÓRIO DE TESTE

QUESTÃO 10)

DOCUMENTAÇÃO DA API:

Está presente no arquivo **documentacaoApi.md**

VALORES A SEREM TESTADOS:

Listagem de produtos:

A listagem dos produtos só tem dois estados possíveis:

- Sem nenhum produto;
- Com um ou mais produtos;

TESTE	ESPERADO
Sem produtos	[]
Com produtos	Lista com os objetos de produtos

Selecionar produto:

Selecionar um produto também só apresenta duas opções:

- Produto existe no catálogo;
- Produto não existe;

TESTE	ESPERADO
ID valido	Objeto do produto
ID invalido	{"detail":"Not found."}

Inserir produto:

Para adicionar um produto as condições ficam mais interessantes:

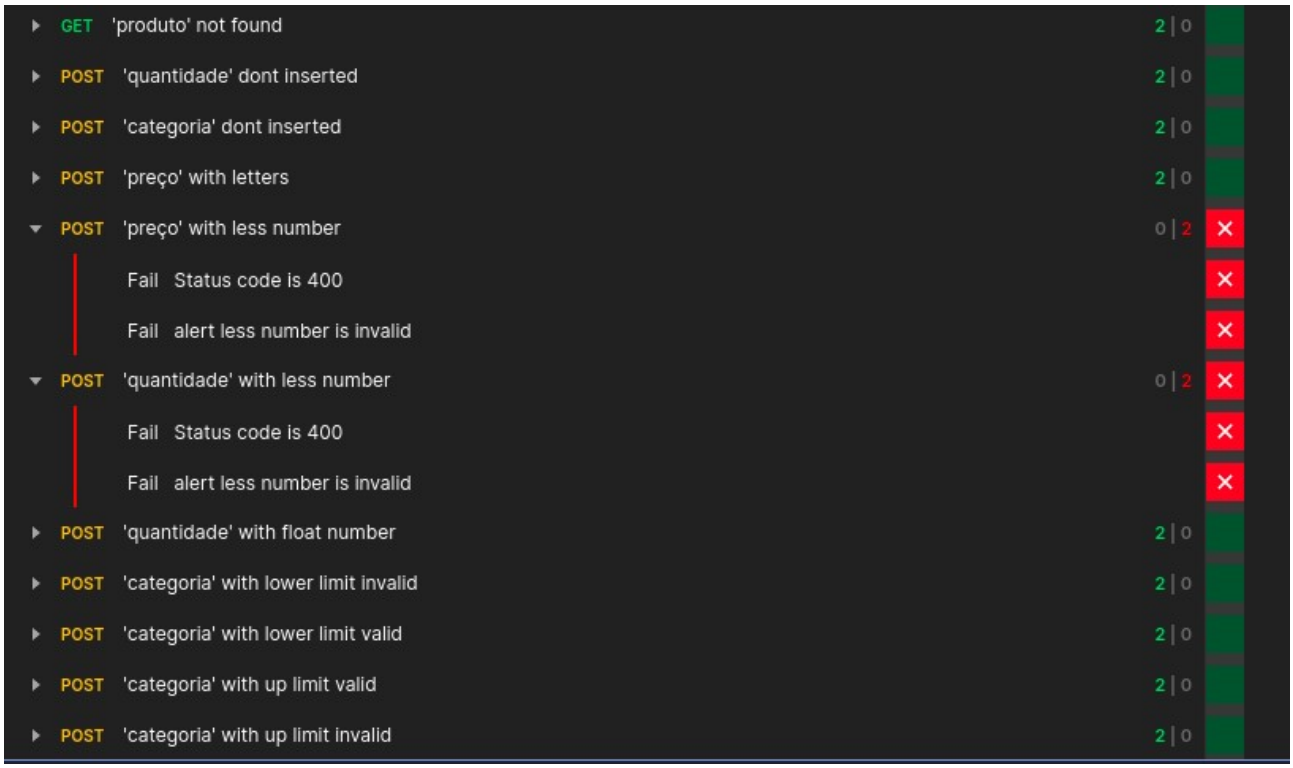
- Os dois primeiros tipos são obrigatórios, então já podemos testar o que ocorre se não passar eles;
- Os dois últimos não são, então o sistema deve se comportar bem sem eles;
- O **nome** poderíamos testar o que acontece com vazio, com muito grande, e com caracteres especiais, mas nenhuma limitação nos foi apresentada;
- O **preço** precisa ser tratado para ser um decimal valido, e não pode ter um preço negativo;
- A **quantidade** deve ser um inteiro valido, pois não podemos comprar meio produto nem letras;

- A **categoria** deve ser um inteiro no intervalo [1, 10], para isso usaremos suas fronteiras.

TESTE	ENTRADA	ESPERADO
Somente obrigatórios preenchidos	{“nome”: “test1”, “preco”: “2.50”}	SUCESSO
Campo **nome** obrigatório	{“preco”: “2.50”}	{“nome”: [“This field is required.”]}
Campo **nome** não vazio	{“nome”: “”, “preco”: “2.50”}	{“nome”: [“This field may not be blank.”]}
Campo **preço** obrigatório	{“nome”: “test4”}	{“preco”: [“This field is required.”]}
Campo **preço** não vazio	{“nome”: “test5”, “preco”: “”}	{“nome”: [“A valid number is required.”]}
Tudo preenchido	{“nome”: “test4”, “preco”: “2.50”, “quantidade”: 1, “categoria”: 1}	SUCESSO
Tudo preenchido, menos **quantidade**	{“nome”: “test5”, “preco”: “2.50”, “categoria”: 1}	SUCESSO
Tudo preenchido, menos **categoria**	{“nome”: “test6”, “preco”: “2.50”}	SUCESSO
Preço com letras	{“nome”: “test7”, “preco”: “aaa”}	{“preco”: [“A valid number is required”]}
Preço com número negativo	{“nome”: “test8”, “preco”: “-5”}	{“preco”: [“A valid positive number is required”]}
Quantidade negativa	{“nome”: “test9”, “preco”: “aaa”, “quantidade”: -1}	{“quantidade”: [“A valid positive number is required”]}
Quantidade inteira	{“nome”: “test10”, “preco”: “aaa”, “quantidade”: 1.1}	{“quantidade”: [“A valid integer is required”]}
Categoria limite inferior invalido	{“nome”: “test11”, “preco”: “2.50”, “categoria”: 0}	{“quantidade”: [“\”0\” is not a valid choice.”]}
Categoria limite inferior valido	{“nome”: “test12”, “preco”: “2.50”, “categoria”: 1}	SUCESSO
Categoria limite superior invalido	{“nome”: “test13”, “preco”: “2.50”, “categoria”: 11}	{“quantidade”: [“\”11\” is not a valid choice.”]}
Categoria limite superior valido	{“nome”: “test14”, “preco”: “2.50”, “categoria”: 10}	SUCESSO

RESULTADOS OBTIDOS:

No arquivo **log/store-api.postman_test_run.json** pode ser encontrado um logo do conjunto de teste realizado, nele vemos que apenas quem falhou foi as análises de quantidade negativa e quantidade inteira:



Valores negativos e valores com casas decimais não estão sendo tratados.