**🧪 Testes no Postman – Validação de Requisições**

🔸 1. **Status code: Code is 200**  
**O que faz:** Verifica se o status final do código é 200.  
**Exemplo:**

pm.test("Status code is 200", *function* () {

    pm.response.to.have.status(200);

});

**Print:**





🔸 2. **Response body: Contains string**  
**O que faz:** Verifica se um determinado valor de texto contém uma substring específica.

**Exemplo:**

pm.test("Body matches string", *function* () {

    pm.expect(pm.response.text()).to.include("available");

});  
**Print:**





🔸 3. **Response body: JSON value check**  
**O que faz:** Valida se um valor dentro de um JSON é igual ao informado.  
**Exemplo:**

pm.test("Your test name", function () {

    var jsonData = pm.response.json();

    pm.expect(jsonData.name).to.eql("doggie");

});  
**Print:**

****



🔸 4. **Response body: Is equal to a string**  
**O que faz:** Verificação automática (assertiva) usada para testar se o corpo da resposta HTTP é exatamente igual a uma string específica.  
**Exemplo:**

pm.test("Body is correct", function () {

    const expectedBody = {

        id: 12,

        category: {

            id: 0,

            name: "string"

        },

        name: "Julio",

        photoUrls: [

            "string"

        ],

        tags: [

            {

                id: 0,

                name: "string"

            }

        ],

        status: "available"

    };

    const actualBody = pm.response.json();

    pm.expect(actualBody).to.eql(expectedBody);

});

**Print:**

****

****

🔸 5. **Response headers: Content-Type header check**  
**O que faz:** Verifica se o Content-Type está presente na resposta da solicitação.  
**Exemplo:**

pm.test("Content-Type is present", *function* () {

    pm.response.to.have.header("Content-Type");

});  
**Print:**





🔸 6. **Response time is less than 200ms**  
**O que faz:** Valida se o tempo de resposta é menor que 200ms  
**Exemplo:**

pm.test("Response time is less than 200ms", *function* () {

    pm.expect(pm.response.responseTime).to.be.below(200);

});  
**Print:**

****



🔸 7. **Status code: Successful POST request**  
**O que faz:** Verifica se a solicitação de POST foi concluída com sucesso  
**Exemplo:**

pm.test("Successful POST request", *function* () {

    pm.expect(pm.response.code).to.be.oneOf([200, 201, 202]);

});  
**Print:**

****



🔸 8. **Status code: Code name has string**  
**O que faz:** Verifica se o nome do status code HTTP da resposta contém uma string específica.  
**Exemplo:**

pm.test("Status code name contains 'OK'", *function* () {

*const* statusText = pm.response.status;

    pm.expect(statusText).to.include("OK");

});  
**Print:**

 

🔸 9. **Response body: Convert XML body to a JSON Object**  
**O que faz:** Converte o corpo da resposta em XML para um objeto JSON manipulável em JavaScript, usando xml2Json.  
**Exemplo:**

pm.test("Status is placed", *function* () {

*const* jsonData = pm.response.json();

    pm.expect(jsonData.status).to.eql("placed2");

});  
**Print:**





🔸 10. **Use Tiny Validator for JSON data**  
**O que faz:** Validar JSON contra um esquema  
**Exemplo:**

const schema = {

"type": "object",

"properties": {

"id": { "type": "integer" },

"petId": { "type": "integer" },

"quantity": { "type": "integer" },

"shipDate": { "anyOf": [

{ "type": "string", "format": "date-time" },

{ "type": "null" }

]

},

"status": { "type": "string" },

"complete": { "type": "boolean" }

},

"required": ["id", "petId", "quantity", "status", "complete"]

};  
**Print:**







**🔧 Variáveis no Postman**

🔸 11. **Get a global variable**  
**O que faz:** Retorna o valor atual de uma variável global definida para todos.  
**Exemplo:** const globalVar = pm.globals.get("var\_name");

🔸 12. **Get a collection variable**  
**O que faz:** Retorna o valor de uma variável de coleção, que é definida apenas para uma coleção específica. **Exemplo:** const collectionVar = pm.collectionVariables.get("apiKey");

🔸 13. **Get an environment variable**  
**O que faz:** Retorna o valor de uma variável do ambiente atual.  
**Exemplo:** const envVar = pm.environment.get("baseUrl");

🔸 14. **Get a variable**  
**O que faz:** Retorna o valor de uma variável, procurando na seguinte ordem de prioridade: local → environment → collection → global.  
**Exemplo:** const anyVar = pm.variables.get("usuario");

🔸 15. **Set a global variable**  
**O que faz:** Define ou atualiza uma variável global.  
**Exemplo:** pm.globals.set("token", "abc123");

**🧼 Limpar Variáveis no Postman**

🔸 16. **Clear a collection variable**  
**O que faz:** Remove uma variável de coleção.  
**Exemplo:** pm.collectionVariables.unset("apiKey");

🔸 17. **Clear an environment variable**  
**O que faz:** Remove uma variável do ambiente atual.  
**Exemplo:** pm.environment.unset("baseUrl");

🔸 18. **Clear a local variable**  
**O que faz:** Remove uma variável local (temporária, válida só na execução atual).  
**Exemplo:** pm.variables.unset("tempVar");

**🔁 Fluxo de Requisição**

🔸 19. **Send an HTTP request**  
**O que faz:** Envia uma requisição HTTP manualmente a partir de um script (útil para chamadas encadeadas ou testes dinâmicos).  
**Exemplo:**

pm.sendRequest("https://api.publicapis.org/entries", function (err, res) {

console.log(res.json());

});