Postman: Fundamentos, Requisições, Variáveis e Automação

Introdução ao Postman

O **Postman** é uma ferramenta popular de **API Client**, utilizada para testar e documentar APIs RESTful e SOAP. Ele permite que desenvolvedores e QAs criem, organizem e compartilhem requisições HTTP, configurem variáveis, testem respostas e automatizem fluxos.

---

1. Fundamentos de uma Requisição HTTP

Uma requisição HTTP é composta por:

1.1 Métodos HTTP

- `GET` — Busca informação.
- `POST` — Envia dados para criar um recurso.
- `PUT` — Atualiza um recurso existente.
- `PATCH` — Atualiza parcialmente um recurso.
- `DELETE` — Remove um recurso.

1.2 Componentes da requisição

- **URL (Endpoint):** Endereço da API.
- **Headers:** Informam tipo de conteúdo, token de autenticação etc.
- **Params:** Parâmetros query passados na URL (?page=1).
- **Body:** Corpo da requisição, usado em POST, PUT, PATCH (JSON, XML, form-data).

1.3 Resposta HTTP

- **Status Code**: Ex: 200 (OK), 404 (Not Found), 500 (Erro interno).
- **Body:** Dados retornados (JSON, XML, texto).
- **Headers:** Informam tipo de resposta, CORS, etc.

---

2. Criando e Organizando Requisições

2.1 Criando uma requisição

1. Clique em "+ New Request".
2. Escolha o método HTTP.
3. Informe o endpoint.
4. Configure headers e body.
5. Clique em **Send**.

2.2 Coleções

- Conjunto de requisições organizadas por projeto.
- Permitem salvar scripts, variáveis e testes.

2.3 Workspaces

- Espaços de trabalho colaborativos.
- Podem ser pessoais, de equipe ou públicos.

---

3. Variáveis no Postman

3.1 Tipos de variáveis

Tipo	Escopo
Global	Visível em qualquer lugar
Collection	Válida para uma coleção
Environment	Depende do ambiente
Local	Temporária na requisição
Data	Usada em testes com CSV/JSON

3.2 Criando variáveis de ambiente

1. Clique em **Environments > +**.
2. Defina `key`, `initial value`, `current value`.
3. Use a variável com `{{variavel_nome}}`.

3.3 Exemplo de uso:

```
GET https://api.exemplo.com/users/{{user_id}}
Headers:
Authorization: Bearer {{token}}
```

---

4. Testes Automatizados com Postman

4.1 Scripts

- **Pre-request Script:** executado antes da requisição.
- **Tests:** executado após a requisição.

4.2 Exemplos de testes:

```
pm.test("Status code é 200", function () {
    pm.response.to.have.status(200);
});

pm.test("Resposta contém propriedade 'id'", function () {
    var jsonData = pm.response.json();
    pm.expect(jsonData).to.have.property("id");
});
```

4.3 Testes com variáveis dinâmicas:

```
let json = pm.response.json();
```

```
pm.environment.set("user_id", json.id);
```

4.4 Scripts pré-requisição:

```
let timestamp = new Date().getTime();
pm.environment.set("now", timestamp);
```

---

5. Automação com Collection Runner

5.1 Collection Runner

- Permite executar uma coleção inteira de requisições sequenciais.
- Suporta arquivos `.csv` ou `.json` para simular dados.

5.2 Testes com massa de dados:

1. Clique em **Runner**.
2. Selecione a coleção.
3. Escolha o ambiente.
4. Importe arquivo de dados.
5. Execute.

5.3 Monitoramento

- Configura execução agendada de coleções.
- Útil para testes de regressão e health checks.

---

6. Autenticação e Headers

6.1 Tipos de autenticação suportados:

- No Auth
- Basic Auth
- Bearer Token
- API Key
- OAuth 1.0/2.0

6.2 Header de Autenticação:

```
Authorization: Bearer {{token}}
```

6.3 Extraindo token dinamicamente:

```
let json = pm.response.json();
pm.environment.set("token", json.accessToken);
```

7. Integrações e Boas Práticas

7.1 Integrações com:

- Jenkins / CI-CD

- GitHub
- Swagger/OpenAPI

7.2 Boas Práticas

- Nomeie requisições e variáveis de forma clara.
- Versione coleções.
- Use exemplos para simular respostas.
- Documente os testes usando Markdown.

Conclusão

O Postman é uma ferramenta poderosa não apenas para testar APIs, mas também para colaborar, documentar e automatizar o ciclo de vida de testes. Dominar variáveis, scripts e runners proporciona mais eficiência ao trabalho de QA e desenvolvimento.